

Lei Wang
Michael Segal
Jenhui Chen
Tie Qiu (Eds.)

LNCS 13473

Wireless Algorithms, Systems, and Applications

17th International Conference, WASA 2022
Dalian, China, November 24–26, 2022
Proceedings, Part III

3
Part III

 Springer

Founding Editors

Gerhard Goos

Karlsruhe Institute of Technology, Karlsruhe, Germany

Juris Hartmanis

Cornell University, Ithaca, NY, USA

Editorial Board Members

Elisa Bertino

Purdue University, West Lafayette, IN, USA

Wen Gao

Peking University, Beijing, China

Bernhard Steffen 

TU Dortmund University, Dortmund, Germany

Moti Yung 

Columbia University, New York, NY, USA

More information about this series at <https://link.springer.com/bookseries/558>

Lei Wang · Michael Segal · Jenhui Chen ·
Tie Qiu (Eds.)

Wireless Algorithms, Systems, and Applications

17th International Conference, WASA 2022
Dalian, China, November 24–26, 2022
Proceedings, Part III

Editors

Lei Wang
Dalian University of Technology
Dalian, China

Michael Segal
Ben-Gurion University of the Negev
Beer-Sheva, Israel

Jenhui Chen
Chang Gung University
Taiwan, China

Tie Qiu
Tianjin University
Tianjin, China

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-031-19210-4 ISBN 978-3-031-19211-1 (eBook)
<https://doi.org/10.1007/978-3-031-19211-1>

© The Editor(s) (if applicable) and The Author(s), under exclusive license
to Springer Nature Switzerland AG 2022

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

The 17th International Conference on Wireless Algorithms, Systems, and Applications (WASA 2022) was held in Dalian during November 24–26, 2022. The conference focused on new ideas and recent advances in computer systems, wireless networks, distributed applications, and advanced algorithms that are pushing forward the new technologies for better information sharing, computer communication, and universal connected devices in various environments, especially in wireless networks. WASA has become a broad forum for computer theoreticians, system and application developers, and other professionals in networking-related areas to present their ideas, solutions, and knowledge of emerging technologies and challenges in computer systems, wireless networks, and advanced applications.

The technical program of WASA 2022 consisted of 94 regular papers and 68 short papers, selected by the Program Committee from 265 full submissions in response to the call for papers. All submissions were reviewed by at least 115 Program Committee members in a 115 double blind process. The submissions cover numerous cutting edge topics: cognitive radio networks; software-defined radio and reconfigurable radio networks; cyber-physical systems (CPSs) including intelligent transportation systems and smart healthcare systems; theoretical frameworks and analysis of fundamental cross-layer protocol and network design and performance issues; distributed and localized algorithm design and analysis; information and coding theory for wireless networks; localization; mobility models and mobile social networking; mobile cloud; topology control and coverage; security and privacy; underwater and underground networks; vehicular networks; radar and sonar networks; PHY/MAC/routing protocols; information processing and data management; programmable service interfaces; energy-efficient algorithms; systems and protocol design; operating system and middleware support; algorithms, systems, and applications of the Internet of Things (IoT); and algorithms, systems, and applications of edge computing, etc. In the first place, we would like to express our grateful appreciation for all Program Committee members for their hard work in reviewing all submissions. Furthermore, we would like to give our special thanks to the WASA Steering Committee for their consistent leadership and guidance; also, we would like to extend our gratitude to the the local chairs (Jingang Yu, Zumin Wang, and Jie Wang), the publication chairs (Chi Lin, Lei Shu, Guangjie Han, and Pengfei Wang), the publicity chairs (Zichuan Xu, Haipeng Dai, Zhibo Wang, and Chenren Xu), organizing chairs (Dongsheng Zhou and Zhenquan Qin), and the Web chair (Bingxian Lu) for their remarkable contributions to WASA 2022, ensuring that it was a successful conference. In particular, we wish to express our deepest respect and

thankfulness to all the authors for submitting and presenting their outstanding ideas and solutions at the conference.

November 2022

Lei Wang
Michael Segal
Jenhui Chen
Tie Qiu

Organization

Steering Committee Members

Xiuzhen Susan Cheng
Zhipeng Cai
Jiannong Cao

George Washington University, USA
Georgia State University, USA
Hong Kong Polytechnic University, Hong Kong,
China
The Ohio State University, USA
University of Macau, Macau, China
Illinois Institute of Technology, USA
University of Pittsburgh, USA
Shanghai Jiao Tong University, China

General Co-chairs

Zhongxuan Luo
Peng-Jun Wan
Xingwei Wang

Dalian University of Technology, China
Illinois Institute of Technology, USA
Northeastern University, China

Program Co-chairs

Lei Wang
Michael Segal
Jen-Hui Chen
Tie Qiu

Dalian University of Technology, China
Ben-Gurion University, Israel
Chang Gung University, Taiwan, China
Tianjin University, China

Publicity Co-chairs

Zichuan Xu
Haipeng Dai
Zhibo Wang
Chenren Xu

Dalian University of Technology, China
Nanjing University, China
Zhejiang University, China
Peking University, China

Publication Co-chairs

Chi Lin
Lei Shu
Guangjie Han
Pengfei Wang

Dalian University of Technology, China
Nanjing Agricultural University, China
Hohai University, China
Dalian University of Technology, China

Local Co-chairs

Jingang Yu	University of Chinese Academy of Sciences, China
Zumin Wang	Dalian University, China
Jie Wang	Dalian Maritime University, China

Web Chair

Bingxian Lu	Dalian University of Technology, China
-------------	--

Organizing Co-chairs

Dongsheng Zhou	Dalian University, China
Zhenquan Qin	Dalian University of Technology, China

Program Committee

Ran Bi	Dalian University of Technology, China
Edoardo Biagioni	University of Hawaii at Manoa, USA
Salim Bitam	University of Biskra, Algeria
Azzedine Boukerche	University of Ottawa, Canada
Zhipeng Cai	Georgia State University, USA
Srinivas Chakravarthi Thandu	Amazon, USA
Sriram Chellappan	University of South Florida, USA
Quan Chen	Guangdong University of Technology, China
Xianfu Chen	VTT Technical Research Centre of Finland, Finland
Xu Chen	Sun Yat-sen University, China
Wei Wang	Sun Yat-sen University, China
Songqing Chen	George Mason University, USA
Soufiene Djahel	Manchester Metropolitan University, UK
Yingfei Dong	University of Hawaii, USA
Zhuojun Duan	James Madison University, USA
Luca Foschini	University of Bologna, Italy
Jing Gao	Dalian University of Technology, China
Xiaofeng Gao	Shanghai Jiao Tong University, China
Jidong Ge	Nanjing University, China
Chunpeng Ge	Nanjing University of Aeronautics and Astronautics, China
Daniel Graham	University of Virginia, USA
Ding Wang	Nankai University, China
Ning Gu	Fudan University, China

Deke Guo	National University of Defense Technology, China
Bin Guo	Northwestern Polytechnical University, China
Meng Han	Kennesaw State University, USA
Suining He	University of Connecticut, USA
Zaobo He	Miami University, USA
Pengfei Hu	Shandong University, China
Peng Sun	The Chinese University of Hong Kong, China
Yan Huang	Kennesaw State University, USA
Yan Huo	Beijing Jiaotong University, China
Holger Karl	University of Paderborn, Germany
Donghyun Kim	Kennesaw State University, USA
Hwangnam Kim	Korea University, South Korea
Bharath Kumar Samanthula	Montclair State University, USA
Abderrahmane Lakas	United Arab Emirates University, UAE
Sanghwan Lee	Kookmin University, South Korea
Feng Li	Shandong University, China
Feng Li	Indiana University-Purdue University Indianapolis, USA
Ruinian Li	Bowling Green State University, USA
Wei Li	Georgia State University, USA
Zhenhua Li	Tsinghua University, China
Zhetao Li	Xiangtan University, China
Peng Li	University of Aizu, Japan
Qi Li	Tsinghua University, China
Yaguang Lin	Shaanxi Normal University, China
Zhen Ling	Southeast University, China
Weimo Liu	George Washington University, USA
Jia Liu	Nanjing University, China
Fangming Liu	Huazhong University of Science and Technology, China
Liang Liu	Beijing University of Posts and Telecommunications, China
Hongbin Luo	Beihang University, China
Jun Luo	Nanyang Technological University, Singapore
Liran Ma	Texas Christian University, USA
Jian Mao	Beihang University, China
Bo Mei	Texas Christian University, USA
Hung Nguyen	Carnegie Mellon University, USA
Pasquale Pace	University of Calabria, Italy
Claudio Palazzi	University of Padua, Italy
Chuan Lin	Northeastern University, China

Junjie Pang	Qingdao University, China
Javier Parra-Arnau	University of Ottawa, Canada
Tie Qiu	Tianjin University, China
Ruben Rios	University of Malaga, Spain
Kazuya Sakai	Tokyo Metropolitan University, Japan
Omar Sami Oubbati	University of Laghouat, Algeria
Kewei Sha	University of Houston - Clear Lake, USA
Hao Sheng	Beihang University, China
Bo Sheng	University of Massachusetts Boston, USA
Tuo Shi	Harbin Institute of Technology, China
Tong Liu	Shanghai University, China
Sukhpal Singh Gill	Queen Mary University of London, UK
Junggab Son	Kennesaw State University, USA
Riccardo Spolaor	Shandong University, China
Chunhua Su	University of Aizu, Japan
Violet Syrotiuk	Arizona State University, USA
Guoming Tang	National University of Defense Technology, China
Bin Tang	Hohai University, China
Xiaohua Tian	Shanghai Jiao Tong University, China
Luis Urquiza	Universitat Politècnica de Catalunya, Spain
Tian Wang	Huaqiao University, China
Yawei Wang	George Washington University, USA
Yingjie Wang	Yantai University, China
Zhibo Wang	Zhejiang University, China
Leye Wang	Peking University, China
Wei Wei	Xi'an University of Technology, China
Alexander Wijesinha	Towson University, USA
Mike Wittie	Montana State University, USA
Kaishun Wu	Shenzhen University, China
Xiaobing Wu	University of Canterbury, New Zealand
Wei Xi	Xi'an Jiaotong University, China
Yang Xiao	University of Alabama, USA
Kun Xie	Hunan University, China
Xuan Liu	Hunan University, China
Kaiqi Xiong	University of South Florida, USA
Kuai Xu	Arizona State University, USA
Wen Xu	Texas Woman's University, USA
Lei Yang	The Hong Kong Polytechnic University, China
Panlong Yang	University of Science and Technology of China, China

Changyan Yi	Nanjing University of Aeronautics and Astronautics, China
Wei Yu	Towson University, USA
Dongxiao Yu	Shandong University, China
Sherali Zeadally	University of Kentucky, USA
Deze Zeng	China University of Geosciences, China
Bowu Zhang	Marist College, USA
Yong Zhang	Shenzhen Institutes of Advanced Technology, China
Yang Zhang	Wuhan University of Technology, China
Cheng Zhang	George Washington University, USA
Xu Zheng	University of Science and Technology of China, China
Yanwei Zheng	Shandong University, China
Lu Zhou	Nanjing University of Aeronautics and Astronautics, China
Jindan Zhu	Amazon Web Services, USA
Tongxin Zhu	Southeast University, China
Yifei Zou	Shandong University, China

Contents – Part III

Theoretical Frameworks and Analysis of Fundamental Cross-Layer Protocol and Network Design and Performance Issues

DC-Gossip: An Enhanced Broadcast Protocol in Hyperledger Fabric Based on Density Clustering	3
<i>Zhigang Xu, Kangze Ye, Xinhua Dong, Hongmu Han, Zhongzhen Yan, Xingxing Chen, Duoyue Liao, and Haitao Wang</i>	

A Time Utility Function Driven Scheduling Scheme for Managing Mixed-Criticality Traffic in TSN	20
<i>Jinxin Yu, Changyan Yi, Tong Zhang, Fang Zhu, and Jun Cai</i>	

Distributed and Localized Algorithm Design and Analysis

Distributed Anti-manipulation Incentive Mechanism Design for Multi-resource Trading in Edge-Assistant Vehicular Networks	31
<i>Dongyu Guo, Yubin Zhou, and Shenggang Ni</i>	

Information and Coding Theory for Wireless Networks

Communication Optimization in Heterogeneous Edge Networks Using Dynamic Grouping and Gradient Coding	47
<i>Yingchi Mao, Jun Wu, Xiaoming He, Ping Ping, and Jianxin Huang</i>	

Design on Rateless LDPC Codes for Reliable WiFi Backscatter Communications	59
<i>Sicong Xu, Xin He, Fan Wu, Guiping Lin, and Panlong Yang</i>	

Design of Physical Layer Coding for Intermittent-Resistant Backscatter Communications Using Polar Codes	72
<i>Xing Guo, Binbin Liang, and Xin He</i>	

MEBV: Resource Optimization for Packet Classification Based on Mapping Encoding Bit Vectors	84
<i>Feng Guo, Ning Zhang, Qian Zou, Qingshan Kong, Zhiqiang Lv, and Weiqing Huang</i>	

NT-RP: A High-Versatility Approach for Network Telemetry Based on FPGA Dynamic Reconfigurable Pipeline	96
<i>Deyu Zhao, Guang Cheng, Yuyu Zhao, and Ruixing Zhu</i>	

An Effective Comprehensive Trust Evaluation Model in WSNs 108
Chengxin Xu, Wenshuo Ma, and Xiaowu Liu

Precise Code Clone Detection with Architecture of Abstract Syntax Trees 117
Xin Guo, Ruyun Zhang, Lu Zhou, and Xiaozhen Lu

Multi-view Pre-trained Model for Code Vulnerability Identification 127
Xuxiang Jiang, Yinhao Xiao, Jun Wang, and Wei Zhang

Localization

Discover the ICS Landmarks Based on Multi-stage Clue Mining 139
Jie Liu, Jinfa Wang, Peipei Liu, Hongsong Zhu, and Limin Sun

Mobility Models and Mobile Social Networking

Dynamic Mode-Switching-Based Worker Selection for Mobile Crowd Sensing 155
Wei Wang, Ning Chen, Songwei Zhang, Keqiu Li, and Tie Qiu

A Distributed Simulator of Mobile Ad Hoc Networks 165
Xiaowei Shu, Hao Wang, Zhou Xu, Kaiwen Ning, and Guowei Wu

Social-Network-Assisted Task Selection for Online Workers in Spatial Crowdsourcing: A Multi-Agent Multi-Armed Bandit Approach 178
Qinghua Sima, Yu-E Sun, He Huang, Guoju Gao, and Yihuai Wang

Privacy-Aware Task Allocation Based on Deep Reinforcement Learning for Mobile Crowdsensing 191
Mingchuan Yang, Jinghua Zhu, Heran Xi, and Yue Yang

Information Sources Identification in Social Networks Using Deep Convolutional Neural Network 202
Jiale Wang, Jiahui Ye, Wenjie Mou, Ruihao Li, and Guangliao Xu

Underwater and Underground Networks

MineTag: Exploring Low-Cost Battery-Free Localization Optical Tag for Mine Rescue Robot 213
Xiaojie Yu, Xu Yang, Yuqing Yin, Shouwan Gao, Pengpeng Chen, and Qiang Niu

TSV-MAC: Time Slot Variable MAC Protocol Based on Deep Reinforcement Learning for UASNs 225
Yuchen Wu, Yao Liu, Zhao Zhao, Chunfeng Liu, and Wenyu Qu

Localization for Underwater Sensor Networks Based on a Mobile Beacon	238
<i>Ying Guo, Longsheng Niu, Rui Zhang, Hongtang Cao, and Jingxiang Xu</i>	

Vehicular Networks

Dataset for Evaluation of DDoS Attacks Detection in Vehicular Ad-Hoc Networks	249
<i>Hong Zhong, Fan Yang, Lu Wei, Jing Zhang, Chengjie Gu, and Jie Cui</i>	

Vehicle-Road Cooperative Task Offloading with Task Migration in MEC-Enabled IoV	261
<i>Jiarong Du, Liang Wang, Yaguang Lin, and Pengcheng Qian</i>	

Freshness-Aware High Definition Map Caching with Distributed MAMAB in Internet of Vehicles	273
<i>Qixia Hao, Jiaxin Zeng, Xiaobo Zhou, and Tie Qiu</i>	

A Scalable Blockchain-Based Trust Management Strategy for Vehicular Networks	285
<i>Minghao Li, Gansen Zhao, and Ruilin Lai</i>	

BP-CODS: Blind-Spot-Prediction-Assisted Multi-Vehicle Collaborative Data Scheduling	296
<i>Tailai Li, Chaokun Zhang, and Xiaobo Zhou</i>	

Performance Analysis of Partition-Based Caching in Vehicular Networks	309
<i>Siyuan Zhou, Wei Wu, and Guoping Tan</i>	

PHY/MAC/Routing Protocols

A Service Customized Reliable Routing Mechanism Based on SRv6	321
<i>Peichen Li, Deyong Zhang, Xingwei Wang, Bo Yi, and Min Huang</i>	

PAR: A Power-Aware Routing Algorithm for UAV Networks	333
<i>Wenbin Zhai, Liang Liu, Jianfei Peng, Youwei Ding, and Wanying Lu</i>	

Multi-Channel RPL Protocol Based on Cross-Layer Design in High-Density LLN	345
<i>Jianjun Lei, Tianpeng Wang, Xunwei Zhao, Chunling Zhang, Jie Bai, Zhigang Wang, and Dan Wang</i>	

Routing Protocol Based on Improved Equal Dimension New Information GM(1,1) Model	354
<i>Jian Shu, Hongjian Zhao, and Huanfeng Hu</i>	

Algorithms, Systems, and Applications of Edge Computing

An Asynchronous Federated Learning Optimization Scheme Based on Model Partition	367
<i>Jing Xu, Lei Shi, Yi Shi, Chen Fang, and Juan Xu</i>	
QoE and Reliability-Aware Task Scheduling for Multi-user Mobile-Edge Computing	380
<i>Weiming Jiang, Junlong Zhou, Peijin Cong, Gongxuan Zhang, and Shiyuan Hu</i>	
EdgeViT: Efficient Visual Modeling for Edge Computing	393
<i>Zekai Chen, Fangtian Zhong, Qi Luo, Xiao Zhang, and Yanwei Zheng</i>	
Joint Optimization of Computation Task Allocation and Mobile Charging Scheduling in Parked-Vehicle-Assisted Edge Computing Networks	406
<i>Wenqiu Zhang, Ran Wang, Changyan Yi, and Kun Zhu</i>	
A Secure Authentication Approach for the Smart Terminal and Edge Service ...	419
<i>Qian He, Jing Song, Shicheng Wang, Peng Liu, and Bingcheng Jiang</i>	
End-Edge Cooperative Scheduling Strategy Based on Software-Defined Networks	431
<i>Fan Li, Ying Qiao, Juan Luo, Luxiu Yin, Xuan Liu, and Xin Fan</i>	
Joint Optimization of Bandwidth Allocation and Gradient Quantization for Federated Edge Learning	444
<i>Hao Yan, Bin Tang, and Baoliu Ye</i>	
Federated Learning Meets Edge Computing: A Hierarchical Aggregation Mechanism for Mobile Devices	456
<i>Jiewei Chen, Wenjing Li, Guoming Yang, Xuesong Qiu, and Shaoyong Guo</i>	
QoS-oriented Hybrid Service Scheduling in Edge-Cloud Collaborated Clusters	468
<i>Yanli Ju, Xiaofei Wang, Xin Wang, Xinying Wang, Sheng Chen, and Guoliang Wu</i>	
Deep Reinforcement Learning Based Computation Offloading in Heterogeneous MEC Assisted by Ground Vehicles and Unmanned Aerial Vehicles	481
<i>Hang He, Tao Ren, Meng Cui, Dong Liu, and Jianwei Niu</i>	

Synchronous Federated Learning Latency Optimization Based on Model Splitting	495
<i>Chen Fang, Lei Shi, Yi Shi, Jing Xu, and Xu Ding</i>	
CodeDiff: A Malware Vulnerability Detection Tool Based on Binary File Similarity for Edge Computing Platform	507
<i>Kang Wang, Longchuan Yan, Zihao Chu, Yonghe Guo, Yongji Liu, Lei Cui, and Zhiyu Hao</i>	
Multi-dimensional Data Quick Query for Blockchain-Based Federated Learning	529
<i>Jiaxi Yang, Sheng Cao, Peng Xiangli, Xiong Li, and Xiaosong Zhang</i>	
Joint Edge Server Deployment and Service Placement for Edge Computing-Enabled Maritime Internet of Things	541
<i>Chaoyue Zhang, Bin Lin, Lin X. Cai, Liping Qian, Yuan Wu, and Shuang Qi</i>	
Optimal Task Offloading Strategy in Vehicular Edge Computing Based on Game Theory	554
<i>Zheng Zhang, Lin Wu, and Feng Zeng</i>	
Aerial-Aerial-Ground Computation Offloading Using High Altitude Aerial Vehicle and Mini-drones	563
<i>Esmail Almosharea, Mingchu Li, Runfa Zhang, Mohammed Albishari, Ikhlas Al-Hammadi, Gehad Abdullah Amran, and Ebraheem Farea</i>	
Meta-MADDPG: Achieving Transfer-Enhanced MEC Scheduling via Meta Reinforcement Learning	572
<i>Yiming Yao, Tao Ren, Meng Cui, Dong Liu, and Jianwei Niu</i>	
An Evolutionary Game Based Computation Offloading for an UAV Network in MEC	586
<i>Qi Gu and Bo Shen</i>	
Edge Collaborative Task Scheduling and Resource Allocation Based on Deep Reinforcement Learning	598
<i>Tianjian Chen, Zengwei Lyu, Xiaohui Yuan, Zhenchun Wei, Lei Shi, and Yuqi Fan</i>	
Improving Gaming Experience with Dynamic Service Placement in Mobile Edge Computing	607
<i>Yongqiang Gao and Zheng Xu</i>	

Cooperative Offloading Based on Online Auction for Mobile Edge Computing	617
<i>Xiao Zheng, Syed Bilal Hussain Shah, Liqaa Nawaf, Omer F. Rana, Yuanyuan Zhu, and Jianyuan Gan</i>	
Incentive Offloading with Communication and Computation Capacity Concerns for Vehicle Edge Computing	629
<i>Chenliu Song, Ying Li, Jianbo Li, and Chunxin Lin</i>	
A Dependency-Aware Task Offloading Strategy in Mobile Edge Computing Based on Improved NSGA-II	638
<i>Chunyue Zhou, Mingxin Zhang, Qinghe Gao, and Tao Jing</i>	
Federated Reinforcement Learning Based on Multi-head Attention Mechanism for Vehicle Edge Caching	648
<i>XinRan Li, ZhenChun Wei, ZengWei Iyu, XiaoHui Yuan, Juan Xu, and ZeYu Zhang</i>	
Research on NER Based on Register Migration and Multi-task Learning	657
<i>Haoran Ma, Zhaoyun Ding, Dongsheng Zhou, Jinhua Wang, and ShuoShuo Niu</i>	
Author Index	667

**Theoretical Frameworks and Analysis
of Fundamental Cross-Layer Protocol
and Network Design and Performance
Issues**



DC-Gossip: An Enhanced Broadcast Protocol in Hyperledger Fabric Based on Density Clustering

Zhigang Xu¹, Kangze Ye¹, Xinhua Dong^{1(✉)}, Hongmu Han¹, Zhongzhen Yan¹,
Xingxing Chen¹, Duoyue Liao¹, and Haitao Wang²

¹ School of Computer Science, Hubei University of Technology, Wuhan 430000, China
xhdong@hbut.edu.cn

² GuangDong Provincial Public Security Department, Narcotics Control Bureau,
Guangzhou 510000, China

Abstract. Low transaction efficiency remains one of the primary constraints to the development of permission blockchain. To enhance the communication performance of blockchain, the majority of research focuses on optimizing the local architecture of blockchain and improving consensus. In practice, increasing the block dissemination capability at the network layer can significantly improve transaction efficiency. We find that the redundancy and instability of the gossip protocol as a broadcast method in Hyperledger Fabric have a significant impact on communication performance. In this work, we introduce the idea of density clustering to propose the DC-Gossip broadcast protocol, constructing a stable network architecture with highly dense connectivity for the blockchain network layer. This architecture can effectively reduce the propagation latency and ensure the integrity of the distributed ledger. In our experiments with Fabric, DC-Gossip reduces latency by more than 19% after 40 blocks are propagated in a stable network environment with more than 100 nodes. Moreover, the latency decreases by 14% in a dynamic network under the identical circumstances.

Keywords: Blockchain · Hyperledger fabric · Network clustering · Transaction latency · Broadcast

1 Introduction

As a form of permissioned blockchain, consortium blockchain has steadily become the most generally used type of blockchain nowadays due to their features such as

This work is supported by the National Natural Science Foundation of China under Grant No. 61772180, the Key-Area Research and Development Program of Guangdong Province 2020B1111420002, and the Science and Technology Project of Department of Transport of Hubei Province 2022-11-4-3, and the Innovation Fund of Hubei University of Technology BSQD2019027BSQD2019020 and BSQD2016019. We sincerely thank the anonymous reviewers for their very comprehensive and constructive comments.

the strict access control mechanism and the flexible channel. However, like with other blockchain systems, while the decentralized feature ensures the security and anonymity of transaction execution, the communication performance issue of low transaction efficiency is also a significant constraint on the deployment and development of consortium blockchain. This is mostly because blockchain utilizes a P2P network to ensure its distributed properties and security, and data transmission between nodes involves a huge number of duplicated operations. Certain messages must be sent frequently and passed for an extended period of time in order to reach the tail nodes of network, inevitably causing delay in the message [1].

After several transactions are bundled into blocks in Hyperledger Fabric consortium blockchain system, they are delivered to each peer in the channel to accomplish the task of sharing the ledger data. This mechanism of communication is based on two distributed protocols: consensus and broadcast. Consensus efficiency and network dissemination rate are the two most important elements affecting the efficiency of these two protocols. Among these, the network dissemination rate has an effect on not only the time necessary to generate and propagate blocks over the channel, but also on the speed with which transactions are distributed among consensus nodes. As a result, increasing the dissemination rate of the blockchain network is an extremely effective necessary measure for improving the consortium blockchain communication performance. And, as the number of peers in current consortium blockchain systems grows, the influence of broadcasting on the efficiency of sequencing services and block dissemination becomes more pronounced, which has an increasing impact on communication performance optimization.

In this paper, we focus on Hyperledger Fabric and analyze the problems of randomness and unfairness in its gossip broadcast protocol, as well as its redundant dissemination structure. Then, using gossip as the foundation, a dynamic adaptive and efficient block dissemination structure is constructed by introducing the idea of density clustering algorithm, and other mechanisms in gossip are adjusted to minimize the impact of new peers and inactive peers on the network, culminating in the design of a new high-performance gossip-based broadcast protocol namely DC-Gossip. DC-Gossip is applicable to large-scale blockchain transactions and possesses several desirable characteristics, including universality and practicability.

In summary, the main contributions of this paper are listed below: Firstly based on the gossip protocol, we improve the traditional epidemic algorithm of gossip by integrating the high-density connectivity concept of density clustering, which dramatically decreases dissemination redundancy and significantly reduces block propagation latency. Then by integrating the notion of core objects and sub-clustering in density clustering algorithms, we were able to reduce the payload on more than half of the peers. Finally we adopt a deterministic dissemination structure, which eliminates isolated nodes caused by random dissemination, greatly improves the transmission reliability of the network, and reduces the probability of incomplete dissemination to a negligible level.

The rest of this paper is organized as follows. Section 2 summarizes and discusses several significant research topics and breakthroughs in the field of permissioned blockchain communication performance optimization, as well as the present condition of broadcasting research in blockchain field. And Sect. 3 presents central idea and framework of our research. Then, in Sect. 4, the specific design of each mechanism in DC-Gossip is explained. Section 5 summarizes the performance evaluation results, compares the upgraded broadcast protocol to the original gossip module and other schemes via simulation, and analyzes the experimental data to demonstrate the practicality of DC-Gossip. Finally, in Sect. 6, we summarize our findings and provide a forecast for the study in future.

2 Related Works and Background

2.1 Communication Performance

At present, the predominant research approach for increasing the communication performance of permission blockchain is to optimize the local architecture of blockchain and to enhance the performance and scalability of Byzantine-style fault-tolerant consensus. For instance, in Zhu Li et al. proposed high-performance consortium blockchain architecture [2], the business logic execution module is decoupled from the consensus verification module to simplify the consensus process and increase consensus speed; and the storage of block information is optimized using CouchDB to improve the write performance of system. Spengler et al. employed CouchDB to optimize the storage performance of heterogeneous medical data in Hyperledger Fabric [3], allowing the blockchain to handle more complicated and efficient query operations, hence optimizing the consortium blockchain system's overall throughput and latency. Marson et al. presented the MITOSIS approach [4], which utilizes a cell mitosis-like partitioning strategy to reduce the latency of permissioned blockchain while preserving high scalability via parallel processing. On the other hand, Yi, and colleagues proposed using a new threshold digital signature approach based on the NP-Hard problem to improve the consensus algorithm [5] and used it to create a brand new, more efficient, and secure blockchain system, which is undoubtedly a classic research proposal as well. The network layer, which lies between the data and consensus layers, has received less attention in study than the above two layers. In reality, it can have a significant positive impact on the overall performance and stability of the blockchain by optimizing the network structure and enhancing transmission rate and transmission reliability.

2.2 Broadcast Protocol

As an important distributed protocol in blockchain systems, the broadcast protocol takes the responsibility of building the topology and passing blocks. The broadcast protocol, a crucial distributed protocol in blockchain systems, is in charge of creating the topology and passing blocks. The Hyperledger Fabric uses

the gossip protocol as a means of distributing data throughout the channel mechanism. The Epidemic method, which is used to maintain replicated databases, made the initial proposal for it in 1987 [6]. The way gossip spreads is similar to how an infection spreads: after starting the message, the source node will randomly choose several peer nodes to push it to, and the infected node will continue to push to further peers after receiving it. Repeat the process continually. Until the message is pushed to the back of the network, the procedure is repeated [1]. The pull and recovery mechanisms are activated to get the missing block content for it from inside the organization or from other organizations when a peer enters this network for network reasons or for the first time. However, despite ensuring the dependability of message distribution, this procedure eventually results in message redundancy and raises the processing load on peers. Additionally, if the quantity of objects being pushed is decreased in an effort to lighten the load, it could result in the formation of isolated nodes and the transmission of missing data. When the pull mechanism is frequently activated to address comparable network instability, the propagation latency will eventually increase significantly.

Recently, blockchain performance research centered on broadcast has begun to garner interest. Weifeng Hao et al. designed the BlockP2P dissemination structure [7], which employs the K-Means algorithm to cluster nodes and then a parallel spanning tree broadcast algorithm to achieve fast data dissemination between and within clusters. Subsequent work [8] adds an effective inactive node detection method to further reduce network payload. Elias Rohrer et al. developed Kadcast [9], a new lightweight P2P protocol based on the UDP protocol that not only decreases the communication overhead of P2P networks, but also has exceptional recovery capabilities in the face of packet loss and random and hostile node failures. Meanwhile both of these methods disregard the applicability within permission chains, and each routing node is subject to heavy demand; Nicolae Berendea et al., on the other hand, uses Hyperledger Fabric as a research object [10] and optimizes the rest of the propagation mechanism to increase the effectiveness and fairness of the Fabric broadcast layer. They also replaces the push component in the gossip with the infect-upon-contagion algorithm. But the trustworthiness of nodes is not taken into account by this enhanced gossip, and the construction of the network architecture is not well discussed; Ying-Hao Zhan proposes using a satellite broadcast network for data transmission and consensus tasks [11], rather than the traditional Internet, and proposes an automatic recovery mechanism for the communication problems inherent in satellite broadcasting, which significantly increases the throughput of the blockchain system. However, this broadcast network has some stability issues, and the effectiveness of the propagation efficiency is also impacted by the network restart recovery mechanism. In order to propose an integrated front-end, back-end, and middle-ware blockchain network architecture that increases the resource utilization of the blockchain system and the security of the network, Gokay Saldamli et al. enhanced the Randomized Gossip Protocol [12] with a failure detection system and a self-healing network architecture. Unfortunately, although he optimizes the storage overhead by enhancing the approach, the architecture is unsuitable

for large-scale networks, and the bandwidth overhead is still significant when the surrounding tables are updated.

3 Method Analysis

In a traditional centralized network, the server function is concentrated in a small number of nodes, making it extremely easy to trigger the single point of failure problem, and the server nodes are constrained in terms of the number of concurrent services they can handle due to their hardware configuration. As a result, the blockchain system that prioritizes fairness and security utilizes a P2P network. Each node in a P2P network can both receive and supply services to other nodes. This makes use of the massive endpoint resources and simultaneously addresses the two major limitations of centralized networks [1].

The performance of a blockchain is measured in two ways: transmission rate and transmission reliability [8]. The network layer reflects these two dimensions as network propagation latency and data coverage, respectively. The shorter the propagation latency between peers, the faster the entire transmission rate of blockchain network. The greater the coverage of the block data during network dissemination, the more robust the network state; on the other hand, the more peers to which the block is not completely propagated, the less reliable the network transmission, which is typically caused by the dissemination structure being unstable and the peer inactivation detection mechanism being imperfect.

To ensure that each peer receives the message in P2P blockchain networks, the gossip is typically employed for data transmission. However, in order to implement the idea of contagion in the classic gossip, it is necessary to enhance network connectedness, i.e., the number of neighbor nodes for each peer in the network. Because the processing capacity of each peer is limited, this raises not only the strain on the peers, but also reduces the transmission rate when the network connectivity is excessive. To address this issue, this research recommends using the clustering idea in order to minimize the network width between linked peers while maintaining network connectivity.

Clustering algorithms are widely employed in contemporary blockchain network research, particularly in social networks. Such as Wu et al. constructed the Bitcoin transaction network as an undirected graph and partitioned it using spectral clustering for the goal of interest mining [13]. In a real-world network environment, each peer is deployed on a physical machine. Each machine is located in a different part of the network, and the latency of communication between them is affected by a variety of factors, including physical distance between the machines, hardware performance, and bandwidth. Thus, these peers can be conceptualized abstractly as being mapped in a multidimensional vector space with a propagation latency equal to the Euclidean distance. However, because the effect of propagation latency is eventually restricted, this vector space remains low-dimensional, and so high-dimensional clustering algorithms such as spectral clustering are inapplicable. Other clustering methods, such as K-means, divide the peers into numerous groups and continue clustering inside each cluster if

additional divisions are required. However, peers are typically scattered more consistently and irregularly in this multi-dimensional vector space. As a result, algorithms like K-means have a tough time dividing effective cluster classes in this application setting.

On the other hand, density clustering excels at identifying clusters of any shape and can rapidly and precisely identify outlier points. High-density connected regions can be discovered by density clustering, which enables rapid distribution of blocks from the leader to other peers in channel, including all leaf nodes. It is crucial to mention that the approach taken in this research does not reject outliers when density clustering screens them out; rather, they are kept in the focus to avoid the high latency peers from compromising the network's general stability. This is due to the fact that in practical applications, each peer in the blockchain network frequently represents a specific user. As a result, it is not advised to remove users from the network because doing so would violate the blockchain's fairness and interfere with the regular operation of applications.

On the basis of the foregoing, we introduce the OPTICS algorithm [14] for determining density connected relationships among each peer, and how data propagation through this relationship network can achieve an optimal balance of multiple dimensions of blockchain network performance. Additionally, because Optics is insensitive to the input parameters, it is capable of effectively reducing subjective errors introduced by parameter design. The working mechanism of each phase of DC-Gossip is depicted in Fig. 1. As it illustrates, its operation is divided into five distinct stages: initial clustering, block pushing, peer state changing, requesting recovery, and timed re-clustering. Among them, peer state changing and requesting recovery work in combination to respond the dynamic network of channel and ensure transmission reliability.

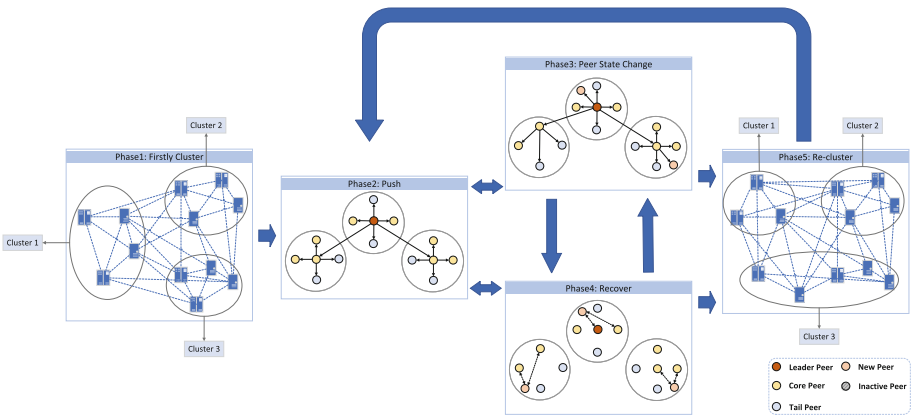


Fig. 1. Overview of the DC-Gossip

4 Design of DC-Gossip

4.1 Initial Dissemination Structure

DC-Gossip treats peers in the channel as a data object in the dataset, and the propagation latency between them is mapped to the Euclidean distance in the Optics vector space for clustering. Note that the clustering is done in terms of the organizations within the channel, and that clustering process of each organization is distinct from the others.

To begin, each organization within the channel elects a leader, which will be responsible for receiving new blocks from the ordering service. The leader is then used as the initial input for density clustering, and through iterative recursion, a large cluster containing leader is identified, and all peers within this cluster are connected via density, i.e., they can be connected directly or indirectly via their respective core objects. Within the large cluster, there are numerous small clusters that correspond to the neighborhoods of each core object. The peers of each neighborhood have a high degree of trust for one another [15], and all block dissemination operations take place preferentially within the neighborhood to which they belong. It is worth noting that the clustering structure generated by our method allows for some neighborhood overlap. Unlike the completely segregated cluster classes generated by algorithms such as partitioning clustering, the neighbors in density clustering are members of the same large cluster without any hierarchical relationship, and the clustering structure is used for data dissemination rather than data comparison and analysis, so the presence of partial overlap has no effect on the operation of broadcast. This is similar to the concept of redundant pushing in the original gossip module, but the difference is that less than half of the peers in the density-connected dissemination structure must perform forwarding duties, and as long as the number of neighboring peers is kept under control, the load on the entire network will be not exceeded.

If there are still peers in the channel that have not been added to this large cluster and are not outliers, a random peer from it is used as the initial input for a new round of iterative identification. As a result of this clustering, one or more large clusters are generated. Each initial input of large cluster is defined as the responsible peer, which is responsible for receiving blocks from leader and propagating them within the cluster during the pushing phase.

The final work is that outliers are accounted for. As previously stated, there are rarely any outlier points in the practical application of Fabric. Thus, each outlier is added sequentially to the neighborhood of the core object with the least delay and also to the large cluster to which the core object belongs in this mechanism. At this point, the structure of the block dissemination is determined. Figure 2(a) illustrates a simple example of cluster class partitioning in the form of a two-dimensional space in which all peers are uniformly distributed in an irregular pattern, all peers are identified as homogeneous cluster relationships, and all peers are divided into multiple neighborhoods by density clustering in order to achieve density reachability among all peers.

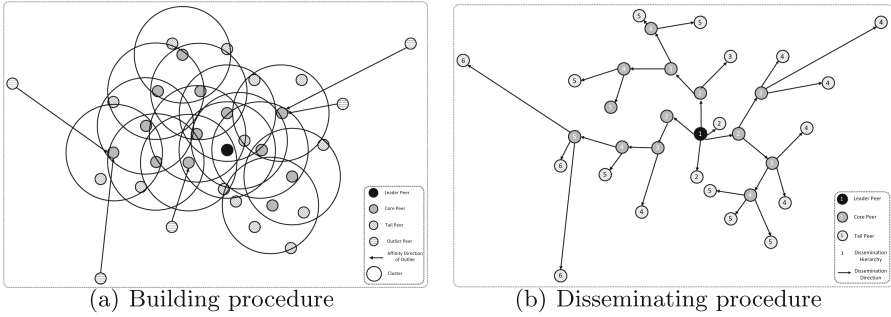


Fig. 2. The density clustering dissemination structure

4.2 Push Dissemination

On the basis of the preceding example, an operational example is assumed to illustrate the block pushing mechanism associated with this dissemination structure: upon receiving a new block from the ordering service, the leader records it to the ledger and deposits it into the buffer. When the buffer is full or the push interval expires, the message is sent to all peers in this neighborhood. These peers evaluate the blocks they receive and, if they have not previously received the same blocks, they repeat the above-mentioned operation of leader. Until all core objects have completed their own pushing work and returned confirmation messages to their superior core objects, and the leader confirms that it has received enough messages, at which point the round of pushing ends. Indeed, this is analogous to the idea of contagion used in the epidemic algorithm [16]. Figure 2(b) depicts a graphical representation of this operational example.

It's worth noting that the method for selecting the leader is typically determined by the requirements of the actual application. If the application prioritizes security and authority of leader, a static election is typically used, in which the user directly designates a peer as the leader; if the application prioritizes fairness, a dynamic election is typically used, in which each peer elects its own leader via some mechanism. When a leader is dropped unexpectedly for special reasons and becomes inactive, the previously selected election scheme is also continued and a new leader is elected.

4.3 Responding to Dynamic Changes of Network

A network cannot always be stable, and peers may be inactive and then reactivated, or additional peers may be added to the channel network. The pull and recovery mechanisms of gossip are intended to address this issue.

Due to the deterministic dissemination structure used in DC-Gossip, the isolated node phenomenon associated with random gossip dissemination is eliminated, and block coverage during the push phase is significantly expanded. As a result, DC-Gossip eliminates the pulling mechanism and retains only the recovery mechanism. Indeed, recovery has no impact on the propagation latencies in a stable network [10].

When a peer goes offline, it is simply marked as inactive and all connections associated with it are deleted; if the peer is a core object, in addition to the above operation, an election is held in its neighborhood to elect a peer to assume all of its responsibilities and become the new core object in that neighborhood; and when the peer rejoins the network, it is treated as a tail peer and rejoins the same cluster to which it was originally connected. And when a new peer requests to join the channel, it is initially assigned randomly to a neighborhood of a cluster. Following a subsequent re-clustering phase, this new peer will be assigned to a cluster that is appropriate for it.

When a peer joins the channel successfully, a recovery mechanism is activated for it. The recovery mechanism of DC-Gossip prioritizes requesting a batch of its missing blocks from the n peers that have the highest trust value with this peer, where n is a user-defined value. If a peer is reconnected and was a member of multiple core objects before inactivation, the peers with the highest trust degree are the core objects of the clusters of which it was a member prior to inactivation and the core objects of the cluster to which it is now a member. This is partly because the smaller the euclidean distance, the greater the trust value, and partly because the probability of missing ledger contents of core objects is lower in comparison to tail peers in the neighborhood, and the ledger contents of multiple core objects are quite reliable when compared together. For newly joined peers and other tail peers that had only one core object prior to inactivation, they can directly request data from the leader, as the trustworthiness of the ledger content of a single core object is poor, and the leader, as the role of receiving blocks directly from orderers, is bound to have the most complete ledger.

4.4 Timed Re-clustering

Since the state of the peers in the network is always changing, if the same dissemination structure is always used, efficiency will inevitably decrease over time. To avoid this solution, periodic re-clustering is required, and DC-Gossip defines a life-cycle du in which the peers that remain online at the end of the du are used to generate a new dissemination structure. A special case must also be considered here: when the du is complete, if any peers are still performing dissemination work, they wait for it to complete before performing re-clustering. If the peer receives a new block during the waiting time, it temporarily stores it in the cache and waits for the new dissemination structure to be determined before pushing the blocks in the cache in an orderly fashion. If a inactive peer applies to rejoin following the determination of the new dissemination structure, it is immediately added to the cluster of a particular core object.

5 Evaluation

This section will focus on evaluating the performance improvement of DC-Gossip in Fabric. We examine the performance differences on various dimensions, such

as block push latency, propagation latency in dynamic networks, and the probability of block incomplete dissemination, between DC-Gossip and the original gossip protocol, as well as the K-Means-based clustering technique described in BlockP2P-EP [8].

5.1 Experimental Setup

Table 1. Experiment basic setup

Bandwidth	Block size	Interval of block sending	Initial number of peers	Latency range	Forwarding delay	Block generation interval	eps	MinPts
5 Mbps	1 kb	15 ms	100	1–30 ms	0–3 ms	15 ms	3	10

This experiment simulates the blockchain network environment under Fabric v2.2 using the *NS-3* network simulator and a network animator named *NetAnim*. The experimental program begins with the deployment of 100 peers, each of which establishes a P2P connection with each other to represent peers belonging to the same channel in the Fabric network.

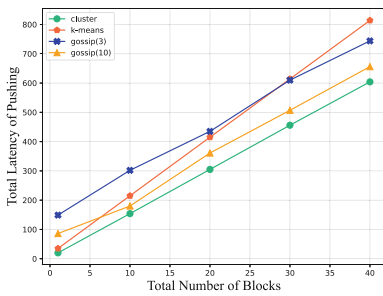
Table 1 summarizes the basic parameters of the simulation experiment, including the network form, bandwidth, block size, block sending interval, initial number of peers, and latency range, etc. The initial number of peers refers to the number of peers in the same channel at the start of the blockchain network, while the latency range refers to the range of propagation latency between peers in the channel, which defines the size of a vector space with latency defined as the euclidean distance. *MinPts* and *eps* are the minimum number of objects in the neighborhood and the radius of neighborhood in Optics respectively. Due to the fact that this experiment is simulated using simulation software, certain parameters cannot be set identically to those on a real Fabric network due to environmental constraints. But that these parameters are based on actual Fabric parameters and the best values are determined through numerous experimental comparisons, it is still possible to observe the relative merits and drawbacks of various approaches based on them.

5.2 Comparison Results

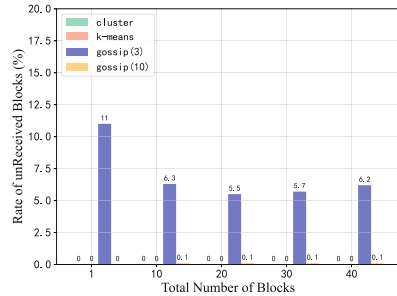
We conduct an experiment during the push phase to compare the performance of various programs under identical conditions. Since a peer pushes a block to random f_{out} peers when it firstly receives one in Fabric’s gossip module, where f_{out} defaults to 3, we created two gossip reference objects in this experiment, one with f_{out} set to 3 and another with f_{out} set to 10.

This experiment establishes two performance metrics: propagation latency and the probability of incomplete dissemination. The performance difference between the three analogues during the pushing phase is depicted in Fig. 3.

As illustrated on Fig. 3(a), the density structure reduces the pushing latency of a single block by a whopping 87% when compared to the original gossip module with the default f_{out} parameter. Additionally, as the number of pushed blocks increases, the two sides maintain a constant latency differential. It has a significant advantage even when compared to a gossip module with f_{out} set to 10. In real-world applications, the f_{out} parameter cannot be set to a large value, such as 10, since each peer is required to perform the push task in the gossip mechanism, otherwise the network will be overburdened. Additionally, the K-Means scheme performs poorly during the push phase, and as transaction duration increases, its overall latency even approaches that of the original Fabric gossip module.



(a) Pushing latency



(b) The probability of incomplete dissemination

Fig. 3. Comparisons in push phase

On the other hand, as illustrated on Fig. 3(b), the probability of incomplete dissemination of original gossip module is always set to a high value when using the default f_{out} parameter. While the probability of incomplete dissemination decreases slightly as the number of block pushes increases, it remains relatively high at around 6%. By contrast, each round of pushing in DC-Gossip ensures that all peers receive the blocks, and the probability of incomplete dissemination is always 0%. At $f_{out} = 10$, the original gossip module also significantly reduces the probability of incomplete dissemination, but again, this is based on the premise of sacrificing load of peers.

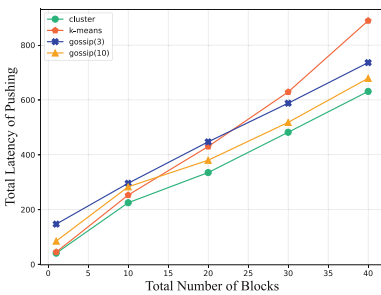
The following set of experiments will compare the propagation performance of each scheme in dynamic networks. To simulate the unstable state of nodes in a real network environment, we set several random time parameters so that every other period, some peers will become inactive or new peers will request to join the network. The requesting peer may be brand-new or reconnected after inactivation. Upon joining the network, the peer will immediately initiate the recovery mechanism and request a batch of missing blocks from others in the channel. The new parameter settings for this experiment are summarized in Table 2, which contains several random parameter value ranges used to simulate

the changes in the network, as well as the number of peers with different dynamic changes for each group comparison.

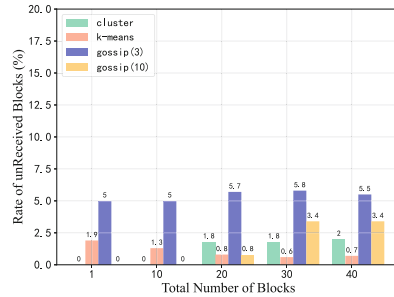
Table 2. Parameter setup for a dynamic network

Number of blocks	Number of newly joined peers	Number of inactive peers	Interval between peers joining	Interval between peers inactivation	Interval between reconnecting
1	3	1	8–15 ms	10–20 ms	15–25 ms
10	15	10	8–15 ms	10–20 ms	15–25 ms
20	20	15	8–15 ms	10–20 ms	15–25 ms
30	25	20	8–15 ms	10–20 ms	15–25 ms
40	30	25	8–15 ms	10–20 ms	15–25 ms

Figure 4 illustrates the performance differences between the schemes as the network dynamics change. As illustrated on Fig. 4(a), as the transaction process lengthens, the network structure is constantly evolving, but a constant gap is maintained between the two methods. For example, even when $f_{out} = 10$, the total latency for the original gossip module remains 40–50 ms longer than DC-Gossip. This demonstrates DC-Gossip mechanism’s superior applicability in dynamic network. But the K-Means scheme still causes high latency in dynamic networks. Figure 4(b) also demonstrates that regardless of the number of blocks in a round of dissemination, block dissemination in the clustering structures which include Optics and K-Means always maintains a very high accuracy, with only a extremely small number of blocks failing to be received by target peers due to transaction conflicts. To ensure comparability, it is worth noting that the original gossip module, like DC-Gossip proposed in this research, omits the pull mechanism and instead relies on the recovery mechanism to reduce the probability of incomplete dissemination caused by dynamic changes in the network.



(a) Propagation latency

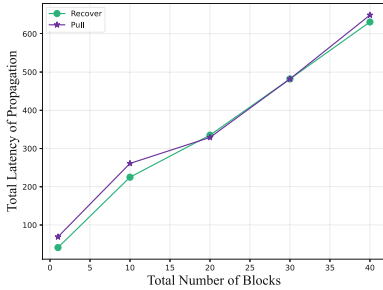


(b) The probability of incomplete dissemination

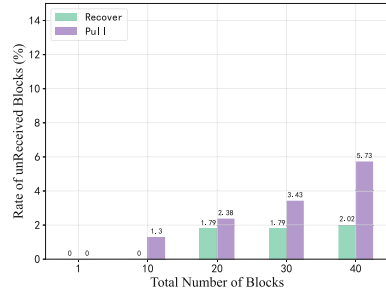
Fig. 4. Comparisons of responding to dynamic network changes

To verify the reasonableness of the trade-off made in our research, we conducted the next experiment. Additionally, this experiment is conducted in a simulated dynamic network environment, with the parameters specified in Tables 1 and 2. This experiment compares the performance of DC-Gossip when using the pull and recovery mechanisms respectively, while maintaining the same start-up times. As illustrated on Fig. 5(a), the difference in propagation latency between the two is negligible, especially as the number of blocks increases, the latency of both mechanism is nearly equal. But Fig. 5(b) demonstrates that the probability of incomplete dissemination with the recovery is significantly lower than the incomplete propagation rate with the pull, particularly when the number of blocks is large. The probability of incomplete dissemination with the recovery remains low at around 2%, while that with the pull continues to climb.

In reality, however, even a low probability of 2% may compromise the ledger’s integrity and traceability within the channel. The reason for this result is that the interval between blocks generation is set to a small value due to the limitation of simulation software, which results in the possibility of transaction conflicts when the same peer receives blocks from multiple parties, and preventing the blocks from being successfully recorded to the ledger. To verify this argument, we increase the interval between blocks generation to 50 ms in simulator, and a further propagation experiment with 40 blocks is conducted using the recovery mechanism with 30 new peers and 25 inactive peers. The probability of incomplete dissemination is found to be reduced to 0.16%, which is an acceptable rate for the blockchain system. Of course, if we want to further reduce the probability of incomplete dissemination, we can keep the pull mechanism in the DC-Gossip, but its activation frequency must be significantly reduced.



(a) Propagation latency



(b) The probability of incomplete dissemination

Fig. 5. Comparisons of pull and recovery in density clustering structure

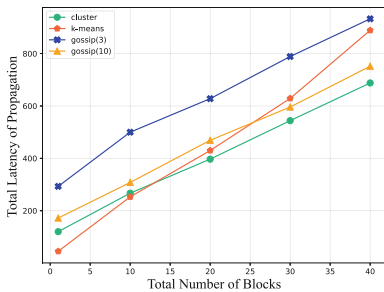
The preceding three experiments are conducted in a vector space with a uniform and dense peer distribution, a distribution that is more representative of the network environment in which blockchains are used in practice. However, the case in which the peers in a channel are distributed across multiple clusters should be considered as well. The fourth experiment simulates the performance

Table 3. Parameter setup for a dynamic network

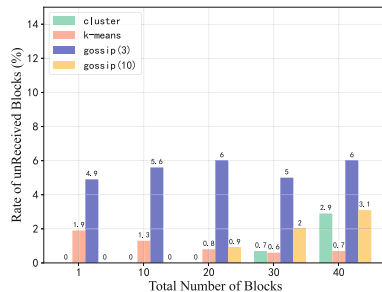
Number of peers	Range of latency	Number of new peers	Number of inactive peers	Number of blocks
100	1–30 ms	15	10	10
150	1–45 ms	20	15	10
200	1–57 ms	25	20	10
250	1–75 ms	30	25	10
300	1–95 ms	35	30	10

disparities between the various schemes when multiple clusters are used. During the process of experiment, it was observed that DC-Gossip divides the initial 100 peers into three clusters using the Optics, and the latency of peers within each intra-cluster ranges from 1–30 ms, while the latency of peers between inter-clusters ranges from 30–60 ms. Additionally, because the distribution of peers is quite different in this experiment than in the previous three, the density clustering parameters are changed to $eps = 5$ and $MinPts = 6$.

By examining the display results on Fig. 6(a), it is clear that DC-Gossip has an overwhelming advantage in terms of propagation latency over the original gossip module in the case multi-cluster dispersion. This is because density clustering is extremely effective at identifying cluster classes and planning a reasonable propagation method accordingly. The gap between the two sides is roughly doubled when compared to the single-cluster case. And as the number of blocks increases, this advantage becomes increasingly apparent. Even when it is used, the latency is reduced by approximately. Concerning the incomplete propagation rate, Fig. 6(b) also demonstrates that DC-Gossip is significantly lower. The reason why the K-Means method is so ineffective is because it is difficult to calculate the aggregated nodes using only the network latency, and the method of network coordinate system [17] and error function to calculate an aggregated



(a) Propagation latency

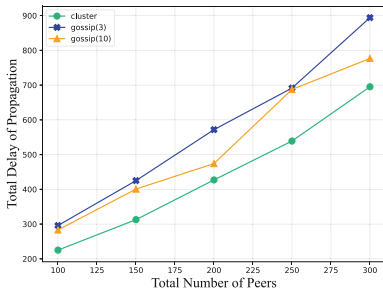


(b) The probability of incomplete dissemination

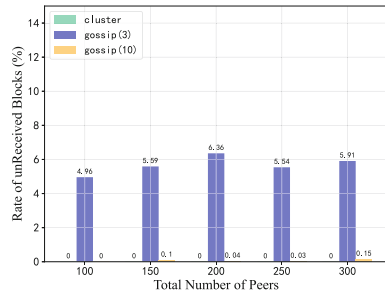
Fig. 6. Comparisons in the case of multi-cluster

node matching function. This strategy can also be tried to be implemented into DC-Gossip in the future.

Each of the preceding four experiment compares the performance of multiple objects in a variety of scenarios using the same number of peers and blocks as the variable. However, the maximum number of peers in a channel is not always 100 in real-world. To demonstrate that the method proposed in this research is also applicable to a larger number of peers, the fifth experiment will compare the performance of the various schemes at various peer counts, which will be using a constant number of blocks and a variable number of peers. Due to the fact that the experimental variables are altered in this experiment, several of the parameter settings listed in Tables 1 and 2 must be altered. Table 3 summarizes the modified parameter settings. It's worth noting that this experiment simulates a situation in which a variable number of peers join the same channel, and in this case, the distribution range of all peers in the vector space cannot be constant. Therefore, the range of propagation latency values between peers is continuously expanded as the total number of peers increases in this set of experiments. In addition, the K-Means scheme was not introduced in this set of experiments to ensure the fairness of the experimental comparison because the number of clusters and the hierarchy of clusters in K-Means had to be manually altered due to the growing size of the network in this set of experiments.



(a) Propagation latency



(b) The probability of incomplete dissemination

Fig. 7. Comparisons in the case of various number of peers

As illustrated on Fig. 7(a), the density clustering broadcast mechanism's optimization capability in terms of propagation latency becomes increasingly apparent as the number of peers in the channel increases. When the number of peers is 100, the propagation latency gap between the clustering structure and the original gossip module is 71 ms; when the number of peers is 300, this gap has increased to 199 ms, which is an increase of 280.8%. This demonstrates the applicability of the law of large numbers in this case: the more peers, the more efficiently a density clustering structure with a similar propagation process to the contagion algorithm can perform. At $f_{out} = 10$ the propagation latency of the original gossip module is unstable and fluctuating, but always within a small

margin of DC-Gossip. As shown on Fig. 7(b), the original gossip module with only the recovery mechanism maintains a high incomplete propagation rate of 5% regardless of the number of peers, but the addition of the pull mechanism inevitably causes it to spend more time disseminating. In comparison, DC-Gossip has no transaction conflicts or missed dissemination at high peer counts and always guarantees complete dissemination at a push count of 10 blocks.

6 Conclusion

To enhance the communication performance of permissioned blockchain and address the issue of inefficient intra-blockchain transactions, we propose a broadcast mechanism named DC-Gossip. By introducing the idea of density clustering, DC-Gossip is capable of constructing a deterministic structure based on the distribution of peers within the current network. When this network structure is used for block dissemination, the latency and the number of invalid propagation can be significantly reduced while maintaining the load balance of each peer.

In terms of theoretical research on network structure construction, the coverage of the algorithms discussed in this paper is insufficient, and subsequent work will focus on exploring other algorithms in machine learning to combine a more efficient broadcasting method. Due to the limitations of the simulation environment, properties such as the life cycle of the network structure could not be demonstrated in the experiments to validate the theory, despite our consideration of various scenarios in block propagation. Subsequent work will concentrate on deploying DC-Gossip in a real Fabric environment in order to confirm the superiority of our method and continue to improve it. We also intend to add a trust value mechanism and incentive mechanism [18] in community detection to increase the security of the network and the reliability of transactions.

References

1. Kan, J., Zou, L., Liu, B., Huang, X.: Boost blockchain broadcast propagation with tree routing. In: Qiu, M. (ed.) *SmartBlock 2018*. LNCS, vol. 11373, pp. 77–85. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-05764-0_8
2. Zhu, L., Yu, H., Zhan, S.X., Qiu, W.W., Li, Q.L.: Research on high-performance consortium blockchain technology. *Ruan Jian Xue Bao/J. Softw.* **30**(6), 1577–1593 (2019). (in Chinese)
3. Spengler, A.C.F., de Souza, P.S.L.: The impact of using couchDB on hyper-ledger fabric performance for heterogeneous medical data storage. In: *2021 XLVII Latin American Computing Conference (CLEI)*, pp. 1–10. IEEE (2021)
4. Marson, G.A., Andreina, S., Alluminio, L., Munichev, K., Karame, G.: MITOSIS: practically scaling permissioned blockchains. In: *Annual Computer Security Applications Conference*, pp. 773–783 (2021)
5. Yi, H., Li, Y., Wang, M., Yan, Z., Nie, Z.: An efficient blockchain consensus algorithm based on post-quantum threshold signature. *Big Data Res.* **26**, 100268 (2021)

6. Demers, A., et al.: Epidemic algorithms for replicated database maintenance. In: Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing, pp. 1–12 (1987)
7. Hao, W., et al.: BlockP2P: enabling fast blockchain broadcast with scalable peer-to-peer network topology. In: Miani, R., Camargos, L., Zarpelão, B., Rosas, E., Pasquini, R. (eds.) GPC 2019. LNCS, vol. 11484, pp. 223–237. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-19223-5_16
8. Hao, W., et al.: Towards a trust-enhanced blockchain P2P topology for enabling fast and reliable broadcast. *IEEE Trans. Netw. Serv. Manage.* **17**(2), 904–917 (2020)
9. Rohrer, E., Tschorsch, F.: Kadcast: a structured approach to broadcast in blockchain networks. In: Proceedings of the 1st ACM Conference on Advances in Financial Technologies, pp. 199–213 (2019)
10. Berendea, N., Mercier, H., Onica, E., Riviere, E.: Fair and efficient gossip in hyper ledger fabric. In: 2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS), pp. 190–200. IEEE (2020)
11. Zhang, Y.-H., Liu, X.F.: Satellite broadcasting enabled blockchain protocol: a preliminary study. In: 2020 Information Communication Technologies Conference (ICTC), pp. 118–124. IEEE (2020)
12. Saldamli, G., Upadhyay, C., Jadhav, D., Shrishrimal, R., Patil, B., Tawalbeh, L.: Improved gossip protocol for blockchain applications. *Cluster Comput.* **25**(3), 1915–1926 (2022)
13. Wu, S.X., Wu, Z., Chen, S., Li, G., Zhang, S.: Community detection in blockchain social networks. *J. Commun. Inf. Netw.* **6**(1), 59–71 (2021)
14. Ankerst, M., Breunig, M.M., Kriegel, H.-P., Sander, J.: Optics: ordering points to identify the clustering structure. *ACM SIGMOD Rec.* **28**(2), 49–60 (1999)
15. Li, J., Liang, G., Liu, T.: A novel multi-link integrated factor algorithm considering node trust degree for blockchain-based communication. *KSII Trans. Internet Inf. Syst. (TIIS)* **11**(8), 3766–3788 (2017)
16. Koldehofe, B.: Simple gossiping with balls and bins. *Stud. Inform. Univ.* **3**(1), 43–60 (2004)
17. Nagpal, R., Shrobe, H., Bachrach, J.: Organizing a global coordinate system from local information on an ad hoc sensor network. In: Zhao, F., Guibas, L. (eds.) IPSN 2003. LNCS, vol. 2634, pp. 333–348. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36978-3_22
18. Shen, M., Duan, J., Zhu, L., Zhang, J., Du, X., Guizani, M.: Blockchain-based incentives for secure and collaborative data sharing in multiple clouds. *IEEE J. Sel. Areas Commun.* **38**(6), 1229–1241 (2020)



A Time Utility Function Driven Scheduling Scheme for Managing Mixed-Criticality Traffic in TSN

Jinxin Yu¹(✉), Changyan Yi¹, Tong Zhang¹, Fang Zhu², and Jun Cai³

¹ College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China

{yujinxin, changyan.yi, zhangt}@nuaa.edu.cn

² State Key Laboratory of Mobile Network and Mobile Multimedia Technology, ZTE Corporation, Shenzhen, China

zhu.fang@zte.com.cn

³ Department of Electrical and Computer Engineering, Concordia University, Montréal, QC H3G 1M8, Canada

jun.cai@concordia.ca

Abstract. With the development of the industrial Internet, IEEE Time-Sensitive Networking (TSN) has attracted more and more attentions due to its capability of providing deterministic network performance. Unlike most existing studies that only considered a single type of traffic, our work addresses the scheduling problem of mixed-criticality traffic in TSN. Time utility function (TUF) is a utility curve that measures the quality of service (QoS) of streams with respect to end-to-end delays. In this paper, we introduce a variety of TUFs for different streams in TSN according to specific timing requirements. To match the transmission protocol of TSN, we first categorize mixed-criticality traffic into periodic and aperiodic streams, and then design a novel scheduling scheme aiming to maximize the total TUF value of all streams. We compare our proposed scheme with two benchmark schemes, and evaluation results show that our proposed one outperforms the counterparts, especially under the worst-case network settings.

Keywords: Time sensitive network · Mixed-criticality traffic scheduling · Time utility function · QoS

1 Introduction

Time-sensitive networking (TSN, IEEE Std 802.1Q [1]) is an emerging network technology, designed for providing the deterministic network performance with low latency and high reliability, and thus it has been more and more widely used in the field of industrial Internet. TSN is a collection of sub-standards. For example, time-aware shaper (TAS, IEEE Std 802.1Qbv) is particularly for deterministic transmission control by configuring the open and close states of the gate

control list in an offline manner. Credit-based shaper (CBS, IEEE Std 802.1Qav) is for preventing low-priority traffic from starving through bandwidth reservation. In addition, frame preemption (IEEE Std 802.1Qbu) improves scheduling performance by allowing high-priority frames to preempt low-priority ones, and frame replication (IEEE Std 802.1CB) is for reducing the packet loss of frame transmissions.

For applications with mixed-criticality traffic, TSN is gradually replacing existing transmission protocols and becomes a new generation of unified standards. One main challenge of scheduling mixed-criticality traffic is that each stream has a specific timing requirement, leading to different utility values per unit of delay. This necessitates to introduce a measurement index to comprehensively characterize the relationship between streams' end-to-end delays and utility values. Time utility function (TUF) [2], which records the utility value of each stream generated by the application from the talker to the listener, has thus been introduced. It is worth noting that there are two kinds of fundamentally different traffic in TSN, i.e., hard real-time (HRT) and soft real-time (SRT), depending on whether there is a strict deadline for the end-to-end delay. Hence, well designed TUFs should be able to depict these features.

In practice, according to traffic arrival patterns, mixed-criticality traffic may be time-triggered (TT) or event-triggered (ET). TT traffic is commonly known as periodic, while ET traffic is often regarded as aperiodic. The periodic streams are offline schedulable, and there has been a lot of research on managing such traffic in TSN [3–6]. For scheduling aperiodic traffic, the recent work [7–11] introduced the new transport mechanism. Nevertheless, most existing studies only consider to optimize the performance of either periodic or aperiodic traffic in TSN, while few consider to optimize both performance jointly.

In this paper, we propose a novel scheme to jointly address the transmission scheduling problem of mixed-criticality traffic consisting of both periodic and aperiodic streams, which can be either HRT or SRT. Particularly, we model the TUF of each stream as a specific non-increasing function characterizing HRT or SRT. Then, for maximizing the total TUF value of all streams, we propose a scheduling scheme integrating a novel sieving strategy for periodic streams and an approximate online algorithm for aperiodic streams.

The main contributions of this paper are summarized in the following.

- We optimize the traditional network constraints in TSN, which takes into account different forms of TUFs for both periodic and aperiodic streams.
- Under the condition of limited transmission capacity, we design a novel sieving strategy for scheduling periodic streams, and then propose an approximately optimal algorithm for scheduling aperiodic streams in an online manner.
- We apply constraint programming solver in the implementation for producing the corresponding scheduling decisions. Then, numerical simulations are conducted to show the superiority of the proposed scheduling scheme over counterparts.

The rest of this paper is organized as follows: Section 2 presents the system model and problem description. In Sect. 3, a novel scheduling scheme for

managing mixed-criticality traffic transmissions is proposed. Simulation results are shown in Sect. 4, followed by conclusions in Sect. 5.

2 System Model and Problem Description

In this paper, we model the TSN network as a directed graph $G(V, L)$, where the node set V consists of TSN switches (SW) and end devices (ED), the link set L includes the full-duplex communication links connecting all nodes. For example, $[v_a, v_b] \in L$ indicates the link from node v_a to node v_b .

In the application level, we consider that each task is generated by a talker's CPU, and the start time of its transmission process is the instant when the first bit of its first stream enters the TSN domain, and this transmission process ends when the last bit of the last stream leaves the TSN domain. After receiving all streams, the listener hands over the data to the upper layer for certain application purposes. Figure 1 shows the transmission process of streams between end devices. Each stream is associated with a specific TUF (either HRT or SRT), whose value is calculated with respect to the end-to-end delay.

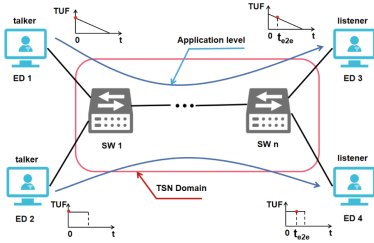


Fig. 1. Transmission process in the application level.

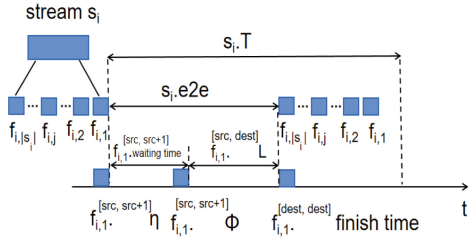


Fig. 2. An illustration of fragmented frames of a stream.

In the TSN domain, information contained in the application is transmitted in units of streams. We denote the input streams set by S . A stream $s_i \in S$ is defined by a tuple $\langle s_i.TUF, s_i.e2e, s_i.D, s_i.T, s_i.Size \rangle$, where $s_i.TUF$ and $s_i.e2e$ represent the time utility function and end-to-end delay of the stream, respectively; $s_i.D$ denotes the time instant when the value of $s_i.TUF$ turns to zero; $s_i.T$ and $s_i.Size$ are the period and length of the stream, respectively. We define $s_i.T = 0$ if stream s_i is aperiodic.

Each stream is further fragmented into multiple frames, i.e., $s_i = \{f_{i,1}, f_{i,2}, \dots, f_{i,j}, \dots, f_{i,|s_i|}\}$, $\forall s_i \in S, |s_i| = \lceil \frac{s_i.Size}{\tau} \rceil$, where τ is the frame size of stream s_i . The j th frame transmission instance of stream s_i from v_a to v_b is represented by $f_{i,j}^{[v_a, v_b]}$, and each of them is defined by a tuple $\langle f_{i,j}^{[v_a, v_b]}.Size, f_{i,j}^{[v_a, v_b]}. \eta, f_{i,j}^{[v_a, v_b]}. \phi, f_{i,j}^{[v_a, v_b]}.L \rangle$, where $f_{i,j}^{[v_a, v_b]}.Size$ and $f_{i,j}^{[v_a, v_b]}. \eta$ separately represent the frame's length and arrival time; $f_{i,j}^{[v_a, v_b]}.L =$

$\lceil \frac{f_{i,j}^{[v_a, v_b]}.Size}{[v_a, v_b].s} \rceil$ is the transmission time of $f_{i,j}^{[v_a, v_b]}$; and $f_{i,j}^{[v_a, v_b]}. \phi$ represents the time when the frame actually starts to be transmitted from v_a to v_b . Obviously, the period of frame $f_{i,j}$ equals that of stream s_i . Figure 2 illustrates the physical meaning of all introduced parameters and the relationship between stream s_i and each of its fragmented frames $f_{i,j}$.

For the set of a variety of input streams $S = S^p \cup S^a$ with a variety of TUFs, where S^p and S^a denote the subsets of periodic and aperiodic streams, respectively, we are interested in finding an appropriate transmission scheduling scheme for maximizing the total TUF of all streams, i.e.,

$$\max O = \sum_{s_i \in S^p} s_i.TUF(s_i.e2e) + \sum_{s_i \in S^a} s_i.TUF(s_i.e2e).$$

To this end, we are required to determine: how the offset of each fragmented frame, i.e., $f_{i,j}^{[v_a, v_b]}$, is determined either in an online or an offline manner, so as to well control the end-to-end delay performance.

3 Proposed Scheduling Scheme

To schedule mixed-criticality traffic in TSN, we assign all periodic streams into the TT queue, and the reason is that periodic streams can be offline controlled by the gate control list, so that given a series of network constraints, the existing optimization solver can produce a solution that meets all constraints (Section III-A). For periodic streams that have lower TUFs, we can then move them out from the TT queue according to the designed stream scheduling strategy (Section III-B), which reserves time slots for the transmission of aperiodic streams. For aperiodic streams, we design an online transmission algorithm to further improve the total TUF value (Section III-C).

3.1 Stream Scheduling Constraints

Scheduling periodic streams in TSN needs to follow basic network constraints [3]. Suppose there is a stream s_i from talker v_{src} to listener v_{dest} , passing through the intermediate nodes $v_1, v_2, \dots, v_{m-1}, v_m$. Denote the routing path of s_i by $R_{s_i} = [[v_{src}, v_1], [v_1, v_2], \dots, [v_{m-1}, v_m], [v_m, v_{dest}]]$. Thus, for periodic stream set $S^p = \{s_1, \dots, s_n\}$, our objective function becomes

$$\sum_{i=1}^n s_i.TUF(s_i.e2e) = \sum_{i=1}^n s_i.TUF(f_{i,|s_i|}^{[v_m, v_{dest}]}.\phi + f_{i,|s_i|}^{[v_m, v_{dest}]} .L - f_{i,1}^{[v_{src}, v_1]}.\eta),$$

where $(f_{i,|s_i|}^{[v_m, v_{dest}]}.\phi + f_{i,|s_i|}^{[v_m, v_{dest}]} .L)$ represents the time when the last frame completes transmission, and $f_{i,1}^{[v_{src}, v_1]}.\eta$ represents the time when the first frame begins transmission.

The constraints for scheduling periodic streams are as follows:

- 1) Capacity constraint: In a hyperperiod, the total time slots occupied by scheduled periodic streams cannot exceed the maximum time slots capacity C allocated to the periodic traffic.
- 2) Frame constraint: The time instant when a periodic frame starts to be transmitted must be no earlier than its arrival time. At the same time, the frame must complete transmission before the beginning of next period.
- 3) Slack size constraint: In order to prevent aperiodic streams from starving, resulting in excessive loss of TUF value of aperiodic streams, we refer to [8] and introduce slacks between periodic frame transmission. We set the lower bound of slacks to X , where X can be dynamically adjusted according to practical settings. At the same time, the slack size should be no larger than the difference between the frame period and its transmission time.
- 4) Time slots reservation constraint: Any two frames cannot be overlapped when they are transmitted on the same link. Based on this, we reserve the slack size between frames according to constraint 3).
- 5) Frame isolation constraint: In order to avoid the jitter caused by interleaving of frames from different streams when using the same queue, we introduce the frame isolation constraint [3].
- 6) End-to-end constraint: The TUF value of the periodic stream received by the listener must be greater than 0. In other words, the periodic streams must be delivered within their deadlines $s_i.D$.

3.2 Sieving Strategy for Scheduling Periodic Streams

Since the total time slots occupied by periodic streams cannot exceed the specified capacity C (subject to constraint 1)), we have to sieve the periodic streams for maximizing the resulting TUF value.

One possible way is to sort all periodic streams according to their potential utility density (PUD) [12], which measures the average TUF value of each stream from $s_i.\eta$ to $s_i.\eta + s_i.D$. For any periodic stream $s_i \in S^p$, We construct a TUF measurement index of a single stream denoted by $s_i.PUD$, where $s_i.PUD = \frac{\int_{s_i.\eta}^{s_i.\eta + s_i.D} TUF(t) dt}{s_i.D}$. We call the stream sieving strategy based on the PUD value as PUD-based scheduling strategy.

Algorithm 1 is used to calculate the injection time for periodic streams within the $O(n)$ time complexities. The scheduling order of periodic streams is determined by their PUD value (Line 1). Then for each sorted stream, the solver will check whether the stream to be scheduled satisfies the above constraints (Lines 3–4). If the solver returns feasible, the stream will be assigned to the scheduled set; otherwise, the stream will not be scheduled (Lines 5–9). However, sieving streams based on the average TUF value may not be accurate since the value of actual TUF will be affected by the queuing delay during the transmission process, which makes the end-to-end delay and actual TUF value uncertain.

Another possible way is to make the end-to-end delay of the stream deterministic, that is, to avoid queuing delay, we refer to [5] to introduce the no-wait scheduling method. The end-to-end delay of a stream adopting the no-wait

Algorithm 1 PUD-based Scheduling Strategy

Input: Unscheduled periodic stream set $S^p = \{s_1, \dots, s_n\}$
Output: Scheduled periodic stream set $S^{p*} = \{s_1, \dots, s_m\}$ ($m \leq n$)
Initialization: $s_i.\phi = s_i.\eta, i \in [1, n]$ $S^{p*} = \emptyset$

- 1: Sort the PUD value of the stream from high to low
- 2: **for** $s_i \in S^p$ **do**
- 3: Put stream s_i into solver
- 4: Check the satisfaction of constraints: 1),2),3),4),5),6).
- 5: **if** solver returns infeasible **then**
- 6: **Continue**
- 7: **else if** solver returns feasible **then**
- 8: $s_i \rightarrow S^{p*}$
- 9: **end if**
- 10: **end for**
- 11: **return** S^{p*}

scheduling method is equal to the transmission delay of each hop multiplied by the number of hops (we ignore the processing and propagation delay in TSN domain for simplicity). Then, the no-wait scheduling constraint can be given as:

$$s_i.e2e = \sum_{j=1}^{|s_i|} \sum_{k=1}^{hopnum-1} f_{i,j}^{[v_k, v_{k+1}]} .L, \forall s_i \in S^p.$$

In this way, the TUF value of each stream is maximized with certainty because of eliminating the queuing delay, and we sieve the streams based on their own deterministic TUF values from high to low. We call this stream sieving strategy based on the deterministic TUF value as no-wait TUF-based scheduling strategy. Compared to Algorithm 1, the scheduling order of no-wait TUF-based scheduling strategy is determined by deterministic TUF value. Besides, the no-wait constraint is added to solver. However, in the worst case, for example, all periodic streams arrive at the same time, then the no-wait TUF-based scheduling strategy can only reserve one periodic stream, leading to a low utilization efficiency. In order to avoid the aforementioned issues, we use the number of periodic streams that can be scheduled (schedulability) as the selection criterion to choose between two strategies, called joint algorithm, which can reduce the end-to-end delay of streams without losing schedulability in certain cases. The scheduling result of periodic traffic will eventually generate a gate control list (GCL), which records the time slots that periodic streams are allowed to pass.

3.3 Online Algorithm for Scheduling Aperiodic Streams

After completing scheduling of periodic streams, the aperiodic streams are scheduled for transmission in the time slots reserved for them according to GCL. It is not suitable to put aperiodic streams into the TT queue in the TSN switch because of the randomness of their arrival times. We choose to put all aperiodic streams into the AVB (Audio Video Bridge) queue in the TSN switch.

Algorithm 2 ASDT Algorithm

Input: s_i : the stream head of the AVB_j queue, $j \in [1, QueueNum]$

- 1: **for** $j = 1$ to $QueueNum$ **do**
- 2: **if** $AVB_j.cbs \geq 0$ **then**
- 3: **if** $s_i.TUF(t) == 0$ **then**
- 4: Discard s_i from AVB_j ;
- 5: **Continue**;
- 6: **end if**
- 7: $\Delta t = \frac{s_i.Size}{\lfloor \frac{v_a + v_b}{s} \rfloor}$;
- 8: $s_i.TUFloss = s_i.TUF(t + \Delta t)$;
- 9: $AVB_j.TUFloss = \frac{\sum_{k=2}^{AVB_j.streamNum} s_k.TUF(t + \Delta t)}{(AVB_j.streamNum - 1)}$;
- 10: **if** $s_i.TUFloss < AVB_j.TUFloss$ **then**
- 11: $s_i \rightarrow$ end of the AVB_j ;
- 12: **else**
- 13: Transmit s_i ;
- 14: **end if**
- 15: **end if**
- 16: **end for**

To improve the TUF value of aperiodic streams, we propose an aperiodic stream dynamic transmission algorithm, called ASDT, as shown in Algorithm 2. The ASDT algorithm cyclically traverses AVB queues from high to low priorities. In each iteration, ASDT checks the state of the stream at the head of current queue AVB_j . The stream s_i will be discarded from the queue if $s_i.TUF = 0$ (Lines 3–6). Before s_i starts transmission, ASDT precalculates the TUF value of s_i and the average TUF value of the queue except s_i after s_i completes transmission, denoted by $s_i.TUFloss$ and $AVB_j.TUFloss$, respectively. If $s_i.TUFloss$ is smaller than $AVB_j.TUFloss$, the stream s_i will not be allowed to transmit and will be put at the end of AVB_j (Lines 10–11), which reduces the end-to-end delay of the stream with higher TUF values.

4 Experimental Evaluation

In this section, we simulate the transmissions of mixed-criticality traffic with different TUFs in the TSN. We use a solver, called constraint programming, to produce the corresponding optimal scheduling decisions. All algorithms are written in Python and run on a PC with a 2.9 GHz CPU and 8 GB of memory.

Consider line topologies (as shown in Fig. 1) with two TSN switches and four end devices. Six periodic streams and six aperiodic streams are generated by two end devices and sent to the other two end devices, where each stream contains one frame with 1542 Bytes. We choose the linear TUF to represent soft real-time traffics. The initial TUF value of the stream is selected from [1000, 2200] randomly, while the arrival time of all streams is random over [0, 200] microseconds. We call our proposed scheme JA-ASDT, which is a combination

of joint algorithm and Algorithm 2, and we compare JA-ASDT for scheduling mixed-criticality streams with the following two common schemes: SA and NSA. SA adopts the PUD-based scheduling strategy (Algorithm 1) for periodic streams and FIFO for aperiodic streams [8]. NSA also adopts the PUD-based scheduling strategy, but it does not reserve slacks for aperiodic streams [3].

We simulate a number of different transmission scenarios by changing the network settings, and among these scenarios, Fig. 3 illustrates the measured minimum TUF value of streams with respect to different capacity limit of periodic streams. The unit of the capacity we set is exactly the time slot that a stream can pass. As can be seen in Fig. 3(a), the proposed JA algorithm will give priority to the no-wait streams without loss of schedulability. As shown in Fig. 3(b), the JA algorithm not only reserves time slots for aperiodic streams, but also the ASDT algorithm better improves the TUF value of aperiodic streams. Lastly, Fig. 3(c) shows the results of all streams, and it is obvious that the proposed JA-ASDT outperforms the others.

Figure 4 reveals the relationship between TUF value and slack size when the capacity of periodic traffic is severely limited (i.e., four units). Also, the unit of slack size we set is the time slot that an aperiodic stream can pass. As shown in Fig. 4(a), TUF value of periodic streams in SA algorithm will gradually decrease with the increase of slack size due to the queuing delay, and the part where JA is higher than SA is when the no-wait TUF-based strategy takes effect. Figure 4(b) shows the superiority of the ASDT algorithm for aperiodic stream scheduling. In general, Fig. 4(c) shows that the proposed JA-ASDT can always achieve a better performance than that of SA under mixed-criticality traffic.

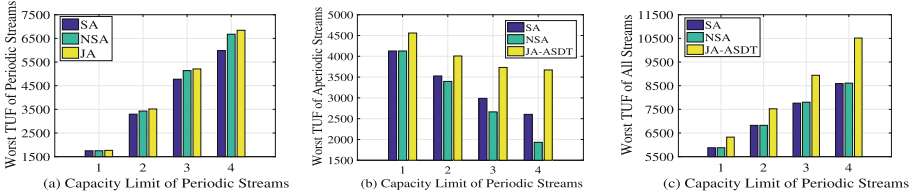


Fig. 3. Worst TUF value with respect to the capacity limit of periodic streams.

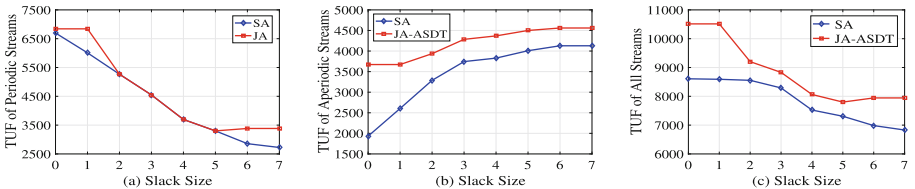


Fig. 4. TUF value with respect to the slack size.

5 Conclusion

In this paper, we design a time utility function (TUF) driven scheduling scheme for managing mixed-criticality traffic in TSN. We consider that mixed-criticality traffic can be categorized into either periodic or aperiodic streams. For maximizing the total TUF value of all streams, a novel scheduling scheme integrating a sieving strategy for scheduling periodic streams and an online algorithm for scheduling aperiodic streams has been proposed. Simulation results show that the proposed scheduling scheme can significantly improve the total TUF value for streams in TSN under the worst-case network settings compared to the state-of-the-art schemes.

Acknowledgement. This work was supported by National Natural Science Foundation of China (NSFC) under Grants 62002164 and 62002165.

References

1. IEEE standard for local and metropolitan area network-bridges and bridged networks. IEEE Std 802.1Q-2018 (Revision of IEEE Std 802.1Q-2014), pp. 1–1993 (2018)
2. Wang, J., Ravindran, B.: Time-utility function-driven switched ethernet: packet scheduling algorithm, implementation, and feasibility analysis. *IEEE Trans. Parallel Distrib. Syst.* **15**(2), 119–133 (2004)
3. Craciunas, S.S., Oliver, R.S., Chmelik, M., Steiner, W.: Scheduling real-time communication in IEEE 802.1 QBV time sensitive networks. In: *Proceedings 24th International Conference RTNS*, pp. 183–192 (2016)
4. Pahlevan, M., Obermaisser, R.: Genetic algorithm for scheduling time-triggered traffic in time-sensitive networks. In: *23rd International Conference ETFA*, vol. 1, pp. 337–344. *IEEE* (2018)
5. Dürr, F., Nayak, N.G.: No-wait packet scheduling for IEEE time-sensitive networks (TSN). In: *Proceedings 24th International Conference RTNS*, pp. 203–212 (2016)
6. Gavriluț, V., Pop, P.: Traffic-type assignment for TSN-based mixed-criticality cyber-physical systems. *ACM Trans. Cyber-physical Syst.* **4**(2), 1–27 (2020)
7. Kim, M., Min, J., Hyeon, D., Paek, J.: TAS scheduling for real-time forwarding of emergency event traffic in TSN. In: *ICTC*, pp. 1111–1113. *IEEE* (2020)
8. Houtan, B., Ashjaei, M., Daneshtalab, M., Sjödin, M., Mubeen, S.: Synthesising schedules to improve QoS of best-effort traffic in TSN networks. In: *29th International Conference RTNS*, 07 Apr 2021, Nantes, France (2021)
9. Yi, C., Cai, J.: Delay-dependent priority-aware transmission scheduling for e-health networks: a mechanism design approach. *IEEE Trans. Veh. Technol.* **68**(7), 6997–7010 (2019)
10. Yi, C., Cai, J., Zhou, S.: A multi-user mobile computation offloading and transmission scheduling mechanism for delay-sensitive applications. *IEEE Trans. Mob. Comput.* **19**(1), 29–43 (2020)
11. Yi, C., Cai, J.: A priority-aware truthful mechanism for supporting multi-class delay-sensitive medical packet transmissions in e-health networks. *IEEE Trans. Mob. Comput.* **16**(9), 2422–2435 (2017)
12. Clark, R.K.: *Scheduling dependent real-time activities*. Carnegie Mellon University (1990)

Distributed and Localized Algorithm Design and Analysis



Distributed Anti-manipulation Incentive Mechanism Design for Multi-resource Trading in Edge-Assistant Vehicular Networks

Dongyu Guo^(✉), Yubin Zhou, and Shenggang Ni

School of Computer Engineering and Science, Shanghai University, Shanghai, China
{guodongyu, z_yubin, nishenggang}@shu.edu.cn

Abstract. In response to the vast and ever-changing task demands of vehicle terminals, the edge-assistant vehicular network (EAVN) supported by the mobile computation offloading (MCO) technic constituted a new paradigm for improving system performance. The existing edge resource trading mechanisms in EAVN were all centralized processing and suffered from several critical drawbacks of the centralized systems, which inspired the research design of distributed trading mechanisms. In this paper, we proposed an efficient distributed reverse combinatorial auction-based trading mechanism under the anti-manipulation check, namely DRCA, to solve the joint multi-task offloading and multi-resource allocation problem in EAVN with overlapping areas, and prevent the participants from manipulating the auction results. We proved that DRCA has achieved the property of faithfulness and analyzed its network complexity. Besides, compared with existing auction-based mechanisms, DRCA could achieve suboptimal social welfare with relatively low system overhead.

Keywords: Edge-assistant vehicular network (EAVN) · Distributed incentive mechanism · Multi-resource allocation

1 Introduction

Over the past decade, the vehicular network was regarded as an effective way to provide innovative services and improve road capacity by connecting vehicles to resource-rich processing servers. Via V2X communication methods, for example, Vehicle-to-Vehicle (V2V), Vehicle-to-Network (V2N), Vehicle-to-Infrastructure (V2I), and Vehicle-to-Pedestrian (V2P), the emergence of 5G-related technologies has brought enticing prospects to vehicular networks [5]. Besides, technological advances have promoted new computation-intensive artificial intelligence applications which will generate massive data in a short time. However, it is difficult for resource-limited vehicles to process massive data separately and also hard to transfer these data to remote servers for real-time analysis. To cope with the vast and constantly changing task demands of vehicles, edge-assistant vehicular network (EAVN) supported by mobile computation offloading technic constitute a

new paradigm for improving system performance [10]. Real-time data processing for onboard tasks is provided by surrounding edge nodes, such as base stations (BSs) and road side units (RSUs). Hence, we found an edge resource trading mechanism in EAVN to meet requirements of RSUs and vehicles.

Traditionally, trading mechanism was controlled by an auctioneer in a **centralized manner** [6, 8, 9]. In effect, centralized auctions in EAVN have following **drawbacks** [4]: (1) The centralized auction required transmitting all the relevant information to a trusted center, but actually the trusted center did not always exist in the real EAVN scenario. (2) The transmission overhead and the computational burden caused on the trusted center were both unmanageable in large-scale EAVN. (3) Most centralized auction mechanisms were not robust. Once the trusted center breaks down, the entire trading system would collapse. To enhance the performance of resource deals, some **distributed incentive mechanisms** had been designed. Due to the heterogeneity of edge resources [1, 13, 15], the probability of multi-task demands [2, 7, 14, 15], the allocation decisions of overlapping areas [13, 14], and possibility of manipulating results [1, 2, 7, 14] were not considered in, mechanisms mentioned above could not be applied to solve our problem. The consistency of the messages, the unreliability of delivery, and the sensitivity to task delay ought to be considered further.

To tackle the above weaknesses, designing a distributed auction-based multi-resource trading system in EAVN has the following **challenges**: (1) It is still a huge challenge for auction designers to handle heterogeneous task demands from vehicles while allocating heterogeneous types of edge resource without global information of overlapping bids. (2) Both communication overhead and computation burden of the entire network need to be carefully managed in mechanism design. (3) In the absence of auctioneers, the distributed auctions were conducted by participants themselves who may take chances to manipulate the auction results through message-passing, information-revelation, and computational actions.

The **contributions** of this paper were summarized as follows: (1) To the best of our knowledge, we are the first to consider the distributed incentive mechanism design for the joint multi-task offloading and multi-resource allocation problem in EAVN. We proposed an efficient distributed reverse combinatorial auction-based multi-resource trading mechanism to solve the NP-hard winner determination problem. (2) The anti-manipulation check was added to the mechanism to resist manipulations of auction result. The mechanism was designed to ensure the property of faithfulness, which means that RSUs would faithfully bid and complete the assigned tasks. (3) The mechanism we proposed led to suboptimal social welfare with acceptable network complexity. In addition, the simulation results of this method showed good performance in terms of social welfare with relatively low system overhead.

2 System Model

Initially, we introduce the main characteristics of the RSUs and the vehicles. Next, to model the interactions between the RSUs and the vehicles, we display a

distributed reverse combinatorial auction framework. Besides, some basic assumptions are presented in detail, and the key parameters of the system are summarized in Table 1.

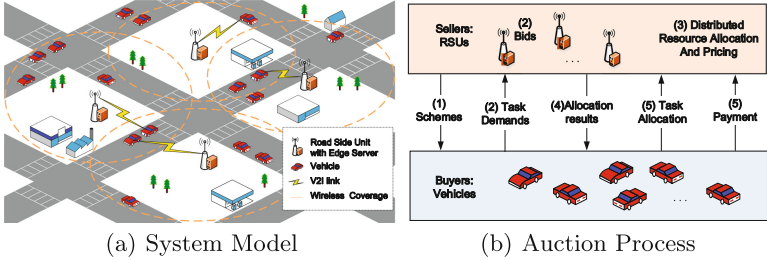


Fig. 1. Illustration of (a) system model and (b) auction process with numerous RSUs and vehicles in EAVN scenario.

2.1 Network Model

As shown in Fig. 1(a), we raise a typical EAVN scenario, which contains a group of geographically distributed RSUs and a set of vehicles, denoted by $\mathcal{I} = \{1, 2, \dots, i, \dots, I\}$ and $\mathcal{M} = \{1, 2, \dots, m, \dots, M\}$, respectively.

On the one hand, to obtain better computation performance, vehicle m with limited local computation capacity f_m want to rent edge resource for a set of computation-intensive tasks \mathcal{T}_m . The “ n th” task from vehicle m is denoted by $T_m^n = \{d_m^n, l_m^n\}$, where d_m^n and l_m^n are the input data size and the number of required CPU cycles, respectively. Noted that one task can only be assigned on one RSU. We use \mathcal{T} to represent the set of all the tasks from vehicles. **On the other hand**, the RSUs, which have much larger resource capacity than the vehicles, offer their edge resources to neighboring vehicles as a set of candidate service schemes $\mathcal{S} = \{1, 2, \dots, s, \dots, S\}$. Each scheme s , contains wireless spectrum and virtual machine (VM) instances, is characterized by four types of resources: bandwidth ($\theta = 0$), CPU frequency ($\theta = 1$), memory ($\theta = 2$), and storage ($\theta = 3$). We use q_θ^s to denote the amount of type θ resource provided in scheme s . For example, the scheme $s = 1$ may consist of 5.8GHz bandwidth, 1.86GHz CPU frequency, 2GB of memory, and 20GB of storage, this scheme can be characterized by $q_0^1 = 5.8$, $q_1^1 = 1.86$, $q_2^1 = 2$, $q_3^1 = 20$. The edge resource of type θ offered by RSU i within its available resource capacity, denoted by C_i^θ .

Edge servers deployed at RSUs receive tasks from neighboring vehicles via vehicle-to-infrastructure (V2I) communication. The set of RSUs that can be accessed by vehicle m is denoted by $\mathcal{I}_m \in \mathcal{I}$. Accordingly, we use $\mathcal{T}_i \in \mathcal{T}$ to denote the set of tasks that can be completely uploaded to RSU i within limited V2I connection time. Furthermore, any two RSUs can communicate with each other directly through wired links.

2.2 Reverse Auction Framework

We model the problem of multi-resource allocation as a distributed reverse combinatorial auction, in which the RSUs are the sellers and the vehicles are buyers.

Table 1. Notations and definitions

Notation	Definition	Notation	Definition
\mathcal{M}, \mathcal{I}	Set of vehicles and RSUs	B_i^j, p_i^j	The RSU i 's " j th" bid and its received payment
\mathcal{T}, \mathcal{S}	Set of tasks and schemes	Q_i^j, c_i^j	Desired tasks and its cost of bid B_i^j
$\mathcal{T}_m, \mathcal{D}_m, \mathcal{I}_m$	Set of vehicle m 's tasks, demands and accessed RSUs	U_m, U_i	Utility of buyer (vehicle) m and seller (RSU) i
$\mathcal{T}_i, \mathcal{B}_i$	Set of RSU i 's achievable tasks and all her bids	U	Joint utility of sellers and buyers (social welfare)
T_m^n, D_m^n, s_m^n	Vehicle m 's " n th" task, demand and scheme	\mathcal{N}_i	Set of seller i 's neighbors
d_m^n, r_m^n	Data size and number of required CPU cycles of T_m^n	$\mathcal{P}_i^j, \mathcal{L}_i^j$	Set of B_i^j 's priority and lagging bids
$q_\theta^s, q_\theta(s_m^n)$	Amount of type θ resource of scheme s and s_m^n	MB_i	Bid-exchange message for bid set \mathcal{B}_i
C_i^θ	Type θ resource capacity of RSU i	MM_i^j	Matching message for bid B_i^j
$\mathcal{W}_b, \mathcal{W}_t$	Set of winning bids and winning task	MP_i^j	Pricing message for bid B_i^j
v_m^n, p_m^n	The utility gain and payment for task T_m^n	MH_i^j	Hypothetical-matching-result message for B_i^j

Assuming that time is slotted, we study the market for one specific time slice. The key parameters of the system are summarized in Table 1.

Buyer. A buyer m is a vehicular user who has a set of computation-intensive tasks \mathcal{T}_m waiting to be executed. Each buyer m can send a set of task demands \mathcal{D}_m to any achievable seller in \mathcal{I}_m . The buyer m 's task demand for task T_m^n can be denoted as $D_m^n = \{T_m^n, s_m^n\}$, where $s_m^n \in \mathcal{S}$ is task T_m^n 's service scheme (mentioned in Sect. 2.1) chosen by the buyer m . Hence, we use $q_\theta(s_m^n)$ to denote the amount of type θ resource needed in scheme s_m^n . If demand $D_m^n \in \mathcal{D}_m$ is fulfilled, task T_m^n will be added to the winning task set \mathcal{W}_t .

The utility gain of offloading task T_m^n consists of two parts, one is the time saved by offloading compared with local computing, and the other is the difference between the local computing energy consumption and the transmission energy consumption:

$$v_m^n = \left(\frac{l_m^n}{f_m} - \left(\frac{d_m^n}{r_m^n} + \frac{l_m^n}{q_1(s_m^n)} \right) \right) \cdot g_m^T + \left(\varphi_m \cdot l_m^n - \frac{p_m d_m^n}{r_m^n} \right) \cdot g_m^E, \quad (1)$$

where r_m^n is the data transmission rate when buyer m is allocated bandwidth $q_0(s_m^n)$, and $q_1(s_m^n)$ is the CPU frequency of chosen scheme s_m^n . Following the similar definitions in [9], we use φ_m to denote the energy consumption per CPU cycle for local computing, and p_m is the transmission power of buyer m . Besides, we use g_m^T (*resp.* g_m^E) to denote the sensitivity of price and task delay (*resp.* energy consumption). The larger the value of g_m^T (*resp.* g_m^E), the higher price buyer m is willing to pay as long as the task delay (*resp.* energy consumption) can be reduced.

The utility of each buyer is the difference between the utility gain and the payment for winning tasks. We use p_m^n to denote buyer m 's payment for task T_m^n . The utility of buyer m can be expressed as:

$$U_m = \sum_{T_m^n \in \mathcal{T}_m \cap \mathcal{W}_t} (v_m^n - p_m^n). \quad (2)$$

Seller. A seller i is an RSU who has idle edge resources that can be leased to buyers, helping to relieve the heavy on-board computation workload while receiving a reasonable payment. Each seller i can submit a maximum of R bids as a set \mathcal{B}_i for different bundles of her desired combinatorial tasks. The " j th" bid from seller i

can be denoted as $B_i^j = \{Q_i^j, c_i^j\}$, where Q_i^j is a subset of \mathcal{T}_i that denotes seller i 's desired tasks, and c_i^j is the claimed operating cost (*i.e.*, computation energy consumption, equipment maintenance and resource management cost). Each seller is "single-minded" and costs c_i^j if and only if she gets the whole desired tasks Q_i^j . Noted that any two bids from one seller cannot overlap. The winning bids in the auction will be added to the winning bid set \mathcal{W}_b .

The utility of each seller is the difference between the payment received from buyers for her winning bids and her claimed cost (which is equal to her private valuation when she bidding truthfully) of the resources allocated to buyers. We use p_i^j to denote seller i 's received payment for bid B_i^j . Thus, the utility of seller i can be formulated as:

$$U_i = \sum_{B_i^j \in \mathcal{B}_i \cap \mathcal{W}_b} (p_i^j - c_i^j). \quad (3)$$

3 Problem Formulation

Our work **objective** is to maximize social welfare with acceptable network complexity by efficient winning bids selection while satisfying sellers' resource capacity constraints, and prevent the sellers from manipulating the auction results. Next, the mathematical formulation of the mechanism design problem is given.

Definition 1 (Social Welfare Maximization (SWM) Problem). *We employ the concept of social welfare to maximize the joint utility (U) of both sellers and buyers when the system is budget balance:*

$$\max U = \sum_{T_m^n \in \mathcal{W}_t} v_m^n - \sum_{B_i^j \in \mathcal{W}_b} c_i^j, \quad (4)$$

$$\text{s.t.} \quad \sum_{B_i^j \in \mathcal{B}_i \cap \mathcal{W}_b} \sum_{T_m^n \in Q_i^j} q_\theta(s_m^n) \leq C_i^\theta (\theta = 0, 1, 2, 3), \forall i \in \mathcal{I}, \quad (5)$$

where constraint (5) indicates that the resources allocated by each seller do not exceed her resource capacity. The SWM problem is proved to be NP-hard by reduction from the NP-complete "INDEPENDENT-SET" problem [11].

Definition 2 (Faithful Distributed Incentive Mechanism Design Problem [12]). *Due to the strategic nature of participants, designing a faithful (anti-manipulation) incentive mechanism requires thorough consideration. A distributed mechanism $d_M = (g, \Sigma, s^M)$ is a faithful implementation of outcome $g(s(\rho))$, when incentive compatibility (IC), communication compatibility (CC), and algorithm compatibility (AC) all hold in a single equilibrium.*

Remark 1 (IC, CC, and AC [4]). The notions of IC, CC and AC in incentive mechanism are as follows: Distributed mechanism d_M is IC (*resp.* CC, AC) if each seller i cannot receive higher utility by deviating from the intended information-revelation (*resp.* message-passing, computational) actions in an equilibrium.

4 Faithful Distributed Multi-resource Trading Mechanism

As shown in Fig. 1(b), without the control of an auctioneer, the faithful *distributed reverse combinatorial auction* (DRCA) mechanism is operated by the rational sellers themselves. The sellers not only compete with each other for rendering resources but cooperate in determining the auction results.

The **process** of DRCA is given as follows: (1) Sellers publish a set of candidate task offloading service schemes \mathcal{S} . (2) Each buyer m sends a set of task demands \mathcal{D}_m to any achievable seller in \mathcal{I}_m . The information of tasks is public and known to every seller. (3) Sellers in \mathcal{I} whose bids of tasks might overlap send messages to each other to determine the winning bids in \mathcal{W}_b and their payments. (4) Sellers announce winning tasks \mathcal{W}_t to buyers. (5) The central bank performs anti-manipulation check and completes the transaction. The winning sellers provides wireless spectrum and VM instances required by tasks.

4.1 The DRCA Mechanism

Let us consider an example to further illustrate the DRCA mechanism. Assuming that 4 RSUs $\mathcal{I} = \{1, 2, 3, 4\}$ bidding for tasks $\mathcal{T} = \{T_1^1, T_2^1, T_3^1, T_4^1\}$ from four vehicles $\mathcal{M} = \{1, 2, 3, 4\}$. The overlapping bids and their characteristics are shown in Table 2. For instance, as shown in the fourth row of Table 2, RSU2 submit only one bid B_2^1 for task T_4^1 and the claimed cost of this bid is 4. The following will introduce four steps to determine the winning bids and their payments.

Table 2. Overlapping bids in example

Bid	Tasks	Cost	Gain	Priority	\mathcal{P}_i^j	\mathcal{L}_i^j	Bid	Tasks	Cost	Gain	Priority	\mathcal{P}_i^j	\mathcal{L}_i^j
B_1^1	$\{T_1^1, T_2^1\}$	4	6	$\sqrt{2}$	\emptyset	$\{B_3^1, B_4^1\}$	B_3^1	$\{T_1^1, T_2^1\}$	5	6	$\sqrt{2}/2$	$\{B_1^1, B_4^1\}$	\emptyset
B_2^1	$\{T_3^1, T_4^1\}$	10	14	$2\sqrt{2}$	$\{B_2^1\}$	$\{B_1^1, B_3^2\}$	B_3^2	$\{T_3^1\}$	4	6	2	$\{B_2^1\}$	$\{B_4^1\}$
B_2^1	$\{T_4^1\}$	4	8	4	\emptyset	$\{B_1^2\}$	B_4^1	$\{T_2^1, T_3^1\}$	8	10	$\sqrt{2}$	$\{B_1^1, B_2^1, B_3^2\}$	$\{B_3^1\}$

Conflict Graph Construction. In order to facilitate the calculation of network complexity, we construct a conflict graph to represent the relationship between sellers in \mathcal{I} : (1) Geographically distributed sellers who has common connected buyers exchange their bid set \mathcal{B}_i with each other by sending a *bid-exchange message* $MB_i = \langle \mathcal{B}_i \rangle$ (Fig. 2(a)). (2) Construct an undirected conflict graph of geographically distributed sellers who has common connected buyers (Fig. 2(b)). In this conflict graph, neighbors are the sellers whose bids overlap. The set of seller i 's neighbors can be denoted as \mathcal{N}_i . (3) Each seller i needs to classify the overlapping bids of bid B_i^j according to their matching priority as follows:

$$\hat{u}_i^j = (v_i^j - c_i^j) / \sqrt{|Q_i^j|}, \quad (6)$$

where $v_i^j = \sum_{T_m^n \in Q_i^j} v_m^n$ is the sum of utility gain in Q_i^j . The bid with higher value of \hat{u}_i^j has higher matching priority. (4) For each bid B_i^j , their overlapping bids can

be divided into two sets: the set of priority bids $\mathcal{P}_i^j = \{B_k^h \mid \hat{u}_k^h \geq \hat{u}_i^j, k \in \mathcal{N}_i\}$, the set of lagging bids $\mathcal{L}_i^j = \{B_k^h \mid \hat{u}_i^j \geq \hat{u}_k^h, k \in \mathcal{N}_i\}$. Noted that if $\hat{u}_i^j = \hat{u}_k^h$ and $i < k$, $B_k^h \in \mathcal{L}_i^j$, if $\hat{u}_i^j = \hat{u}_k^h$ and $k < i$, $B_k^h \in \mathcal{P}_i^j$. For instance, RSU2 can get B_2^1 's priority bid set $\mathcal{P}_2^1 = \emptyset$ and lagging bid set $\mathcal{L}_2^1 = \{B_1^1\}$ by calculating the \hat{u}_i^j of overlapping bid B_1^1 and B_2^1 .

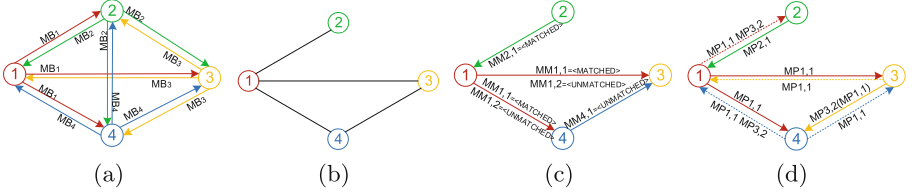


Fig. 2. Illustration of conflict graph and message flow: (a) bid-exchange message flow (b) updated conflict graph (c) matching message flow (d) pricing message flow

Multi-resource Allocation. Sellers determine the allocation results by sending each bid's *matching message* $MM_i^j = \langle \text{MATCHED/UNMATCHED} \rangle$ to their neighbors (Fig. 2(c)). The interactions between sellers are described as follows, which is also shown in Table 3.

Each seller i can only send one matching message MM_i^j for her bid $B_i^j \in \mathcal{B}_i$. If and only if seller i receives all the matching messages about bids in \mathcal{P}_i^j are unmatched and she has sufficient resources left to perform all the task schemes of Q_i^j , she can send $MM_i^j = \langle \text{MATCHED} \rangle$ for bid B_i^j to owners of \mathcal{L}_i^j , otherwise, she sends $MM_i^j = \langle \text{UNMATCHED} \rangle$ to owners of \mathcal{L}_i^j . For example, bid B_2^1 is first matched because it has no priority bids. So RSU2 sends $MM_2^1 = \langle \text{MATCHED} \rangle$ to RSU1 (who owns $B_1^1 \in \mathcal{L}_2^1$). Then, RSU1 sends $MM_1^1 = \langle \text{UNMATCHED} \rangle$ to RSU3 and RSU4. Lastly, the winning bid set is $\mathcal{W}_b = \{B_1^1, B_2^1, B_3^1\}$.

Pricing. Each seller i sends a *pricing message* $MP_i^j = \langle \text{PRICING}, B_k^h \rangle$ to neighbor k ($B_k^h \in \mathcal{L}_i^j$) to determine the payments for winning bids (Fig. 2(d)). Upon receiving MP_i^j of \mathcal{P}_i^h , seller k need to reply a *hypothetical-matching-result message* $MH_k^h = \langle B_i^j, \text{MATCHED/UNMATCHED} \rangle$ to seller k , where the message is the matching result of bid B_k^h if bid B_i^j is not fulfilled. Noted that the matching result of B_k^h 's priority bid $B_\lambda^\mu \in \mathcal{P}_k^h \setminus \{B_i^j\}$ can be indirectly affected by the result of bid B_i^j , seller k also needs to send pricing message MP_i^j to the owner of B_λ^μ . After that, seller k collects all of the hypothetical-matching-result messages about B_i^j and sends MH_k^h to seller i . Sellers utilize the critical-value-based pricing strategy to determine the payments for winning bids. The critical bid for defining the payments of bid B_i^j is the one that loses exactly because of bid B_i^j , *i.e.*, the bid $B_k^h \in \mathcal{L}_i^j$ is fulfilled without bid B_i^j . But if there is no fulfilled bid in \mathcal{L}_i^j , the bid with maximum \hat{u} in \mathcal{L}_i^j would be chosen as B_i^j 's critical bid.

Table 3. Multi-resource allocation and pricing algorithm

Multi-Resource Allocation Algorithm	Pricing Algorithm
Input: $\mathcal{B}_i, \mathcal{P}_i^j, \mathcal{L}_i^j, \mathcal{S}, C_i^\theta$ Output: W_i, W_b $W_i \leftarrow \emptyset, W_b \leftarrow \emptyset, \mathcal{P}_i^j \leftarrow \mathcal{P}_i^j$ foreach $B_i^j \in \mathcal{B}_i$ do if $\mathcal{P}_i^j = \emptyset$ while $\sum_{T_m^n \in Q_i^j} q_\theta(s_m^n) \leq C_i^\theta$ do Send $MM_i^j = \langle \text{MATCHED} \rangle$ to \mathcal{L}_i^j ; $W_i \leftarrow Q_i^j, W_b \leftarrow B_i^j$; $C_i^\theta \leftarrow C_i^\theta - q_\theta(s_m^n)$; else Receive MM_k^h from \mathcal{P}_i^j ; if $MM_k^h = \langle \text{MATCHED} \rangle$ or $\sum_{T_m^n \in Q_i^j} q_\theta(s_m^n) > C_i^\theta$; Send $MM_i^j = \langle \text{UNMATCHED} \rangle$ to \mathcal{L}_i^j ; Break ; else $\mathcal{P}_i^j \leftarrow \mathcal{P}_i^j \setminus \{B_k^h\}$; if $\mathcal{P}_i^j \neq \emptyset$ Send $MM_i^j = \langle \text{MATCHED} \rangle$ to \mathcal{L}_i^j ; $W_i \leftarrow Q_i^j, W_b \leftarrow B_i^j$; $C_i^\theta \leftarrow C_i^\theta - q_\theta(s_m^n)$; Break ; return W_i and W_b ; 	Input: $\mathcal{B}_i, \mathcal{N}_i, \mathcal{P}_i^j, \mathcal{L}_i^j, W_b, v_i^j$ Output: p_i^j $p_i^j \leftarrow \emptyset, \mathcal{P}_i^j \leftarrow \mathcal{P}_i^j, \mathcal{L}_i^j \leftarrow \mathcal{L}_i^j$; foreach $B_i^j \in \mathcal{B}_i \cap W_b$ do Send MP_i^j to \mathcal{L}_i^j ; while $\mathcal{L}_i^j \neq \emptyset$ do Receive MH_k^h from \mathcal{L}_i^j ; if $MH_k^h = \langle B_i^j, \text{MATCHED} \rangle$ $p_i^j = v_i^j - (v_k^h - c_k^h) / \sqrt{ Q_k^h / Q_i^j }$; Break ; $\mathcal{L}_i^j \leftarrow \mathcal{L}_i^j \setminus \{B_k^h\}$; if $\mathcal{L}_i^j = \emptyset$ Find B_λ^μ with maximum \hat{u}_λ^μ in \mathcal{L}_i^j ; $p_i^j = v_i^j - (v_\lambda^\mu - c_\lambda^\mu) / \sqrt{ Q_\lambda^\mu / Q_i^j }$; foreach $B_i^j \in \mathcal{B}_i$ do if Receive MP_k^h from \mathcal{N}_i while $\mathcal{P}_i^j \setminus \{B_k^h\} \neq \emptyset$ do Send MP_k^h to $\mathcal{P}_i^j \setminus \{B_k^h\}$; Receive MH_λ^μ from $\mathcal{P}_i^j \setminus \{B_k^h\}$; if $MH_\lambda^\mu = \langle B_k^h, \text{MATCHED} \rangle$ Reply $MH_i^j = \langle B_k^h, \text{UNMATCHED} \rangle$ to k ; Break ; $\mathcal{P}_i^j \leftarrow \mathcal{P}_i^j \setminus \{B_k^h\}$; if $\mathcal{P}_i^j = \{B_k^h\}$ Reply $MH_i^j = \langle B_k^h, \text{MATCHED} \rangle$ to k ; return p_i^j ;

Thus, the payment of bid B_i^j is exactly the claimed cost at which the transition between B_i^j being before and after B_k^h in the matching order happens [4]:

$$p_i^j = v_i^j - (v_k^h - c_k^h) / \sqrt{|Q_k^h| / |Q_i^j|}. \quad (7)$$

For instance, RSU3, one of the winning sellers, sends $MP_3^2 = \langle \text{PRICING}, B_4^1 \rangle$ to RSU4. RSU4 sends $MP_3^2 = \langle \text{PRICING}, B_1^1, B_1^2 \rangle$ to RSU1, RSU1 also needs to send $MP_3^2 = \langle \text{PRICING}, B_2^2 \rangle$ to RSU2. After that, RSU2 replies $MH_2^1 = \langle B_3^2, \text{MATCHED} \rangle$ to RSU1, RSU1 replies $MH_1^2 = \langle B_3^2, \text{UNMATCHED} \rangle$ and $MH_1^1 = \langle B_3^2, \text{MATCHED} \rangle$ to RSU4. Apparently, B_4^1 still can not be matched if B_3^2 does not exist, so bid B_4^1 with maximum \hat{u} is chosen as the critical bid. Therefore, the payment for B_3^2 is calculated as $p_3^2 = 6 - \sqrt{2} \approx 4.59$. Similarly, $p_1^1 = 5.00, p_1^2 = 5.17$. The pricing rule is shown in Table 3. Furthermore, winning seller i can send MM_i^j, MP_i^j and MH_i^j together to her neighbors.

Anti-manipulation Check. Noted that in a distributed auction, the computation and communication of each seller is responded and confirmed by at least one of her neighbors. Therefore, when clearing a transaction, each seller needs to submit all her interaction messages to the *Credit Clearance Service* (CCS), which can subtly control sellers' manipulated strategies on computation and communication [3]. After all the messages are collected, the CCS checks the messages, authorizes the resource allocation and collects the payment. Besides, the CCS does not always need to have a reliable communication channel with each seller, or participate in the decisions of allocation and pricing.

4.2 Proof of Properties

Theorem 1 (Faithfulness). *DRCA is a faithful distributed mechanism.*

Proof. To prove this theorem, we show that DRCA satisfies centralized IC, CC and AC [12]. **The corresponding centralized auction of DRCA is IC:** Assumed that seller i reports truthful bid $B_i^j = \{Q_i^j, c_i^j\}$ and untruthful bid $\hat{B}_i^j = \{Q_i^j, \hat{c}_i^j\}$. The central auctioneer greedily takes winning bids in an order determined by formula (6). Assume first that B_i^j is a winning bid. As long as \hat{c}_i^j is smaller than c_i^j , the bidder still wins with the same payment, thus misreporting his value would not be beneficial. When \hat{c}_i^j is greater than c_i^j , seller i will lose, gaining zero utility. If B_i^j is a losing bid, its priority value in formula (6) must be smaller than the corresponding critical value, so the payment for any winning bid \hat{B}_i^j will be smaller than truthful cost c_i^j , making this deviation nonprofitable. **DRCA satisfies CC and AC:** The CCS will find the miscalculation of priorities which causes communication chaos, and any illegal match will be caught under the anti-manipulation check. Thus, seller i would follow the intended message-passing and computation strategy.

Network Complexity [4]. We use an interconnection network $\mathcal{G} = (\mathcal{I}, \mathcal{E})$ to represent the conflict graph (Fig. 2(b)), where \mathcal{E} contains all the connected links among the sellers in \mathcal{I} . The network complexity of DRCA is calculated in terms of five metrics. **The maximum number of messages sent over any one link in \mathcal{G} :** In the worst case, every seller submits the maximum number of bids R and every bid's payment needs to be calculated through one link. Hence, there are $(2R|\mathcal{I}| + 2R + 2)$ messages sent on each link (*i.e.* two bid-exchange messages, $2R$ matching messages, $R|\mathcal{I}|$ pricing messages, and $R|\mathcal{I}|$ hypothetical matching result messages). **The total number of messages sent over \mathcal{G} :** In the worst case, the total number of messages sent over \mathcal{G} is $(2R|\mathcal{I}| + 2R + 2)|\mathcal{E}|$. **The maximum size of a message:** The maximum length of any bid-exchange message MB is constant Z bytes. The maximum size of a message is not as long as Z bytes. **The local computational burden at each node:** The toughest part throughout DRCA is the division of neighbors' bids, which takes at most $O(\delta)$ time in the worst case, where δ is the maximum degree of the network. **The storage required at each node:** Each seller is required at most $O(\delta|\mathcal{I}|)$ space to store messages and local outcome in the worst case.

5 Simulation Results

This section presents numerical results to illustrate the validity of DRCA. On the basis of referring to [9,10], the simulation environment is as follows: the EAVN scenario with dimensions of $1200\text{ m} \times 1200\text{ m}$ square region is considered, where 500 vehicles and 100 RSUs are randomly distributed in vehicular network simulator VISSIM. The number of demands for each vehicle is within $[0, 4]$, and the number of desired tasks of each bid is within $[2, 5]$ ($R = 3$). Besides, we design 10 schemes that contain 4 types of resources follows the uniform distribution $[10, 20]$, and each type of resource capacity of RSUs follows the poisson distribution in $[100, 400]$.

This study compared the performance of DRCA with four mechanisms: an upper bound, centralized VCG mechanism that maximizes the social welfare; a distributed version of VCG (namely DVCG [15]) which constructs a depth-first search (DFS) tree to achieve optimal social welfare, satisfying the property of faithfulness with exponential system overhead; a centralized combinatorial auction mechanism (namely OCRAP [8]) which has a similar allocation rule with DRCA to achieve suboptimal social welfare with acceptable system overhead, without satisfying the property of faithfulness; and a centralized mechanism ensuring the fairness of the allocation (namely DOCAT [6]).

5.1 Allocation Efficiency

This paper evaluate the allocation efficiency of DRCA in terms of social welfare, and compare it with three centralized incentive mechanisms (DVCG achieves the same social welfare as VCG [15]). 50 iterations are tested to measure the allocation efficiency with average results.

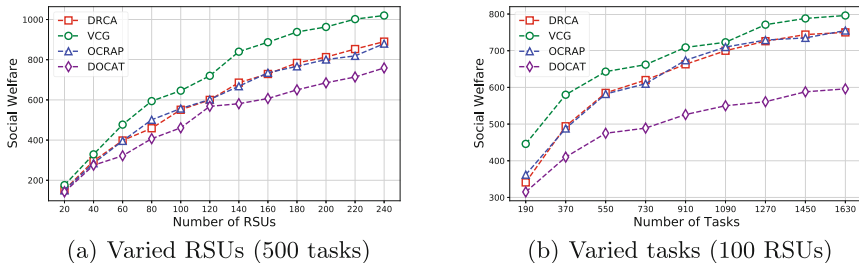


Fig. 3. (a) Social welfare with varied RSUs. (b) Social welfare with varied tasks.

From Fig. 3, we observe that VCG achieves higher social welfare than DRCA under the same parameters. This is because that VCG is designed for optimal allocation, while DRCA is more inclined to greedy policy. DRCA and OCRAP have similar suboptimal social welfare since both of them follow the greedy-based allocation rule and the critical payment rule. DOCAT has the lowest social

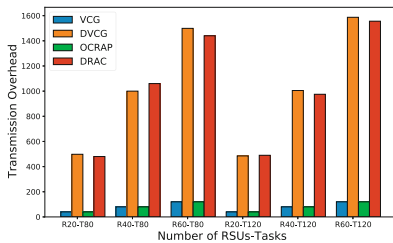
welfare because it sacrifices a portion of social welfare to ensure the fairness of the resource allocation. In addition, when the number of tasks is fixed, we can see that increasing number of RSUs leads to higher social welfare. Meanwhile, social welfare increases with the number of tasks and saturation is reached when the number of tasks is about 1530.

5.2 System Overhead

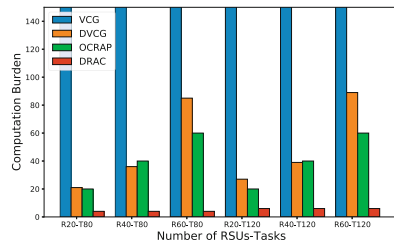
We measure DRCA’s system overhead on account of transmission overhead and computation burden, where transmission overhead is defined as the total number of messages over the network, and computation burden is defined as the average computational burden of all nodes [4].

In Fig. 4(a), the transmission overhead of distributed incentive mechanisms (DVCG and DRCA) and their corresponding centralized incentive mechanisms (VCG and OCRAP) are compared, where “R20-T80” denotes that 20 RSUs bid for 80 tasks. We can find that centralized mechanisms have the least number of messages since they only need basic communications between the auctioneer and other participants. On the contrary, the quantity of messages of distributed mechanisms increases exponentially with the amount of RSUs as each RSU need to enumerate some potential matching results of her constraint view.

In Fig. 4(b), the computation burden of each node in distributed mechanisms is much smaller than that in centralized mechanism, since the computation of the auctioneer in a centralized mechanism is divided among multiple nodes in its corresponding distributed mechanism. In addition, though DRCA and DVCG have similar transmission overhead, the computation burden of DRCA is much smaller than that of DVCG. The reason is that each node in DRCA only calculates the priorities of her neighbors’ bids at the begin of the auction. However, each node in DVCG is required to calculate optimal welfare of its subtree under some potential matching conditions, which makes the computation burden growing exponentially with the number of her neighbors.



(a) Transmission Overhead



(b) Average computation burden

Fig. 4. (a) Number of messages with varied RSUs and tasks. (b) Average computation burden of all nodes with varied RSUs and tasks.

6 Conclusion

In this study, we have considered the distributed incentive mechanism design for the joint multi-task offloading and multi-resource allocation problem in EAVN, and have proposed a faithful distributed auction mechanism under the anti-manipulation check, namely DRCA. Besides, we have analyzed its economic property and network complexity. Compared with existing auction-based mechanisms, DRCA could achieve suboptimal social welfare with relatively low system overhead.

References

1. Baranwal, G., Kumar, D.: DAFNA: decentralized auction based fog node allocation in 5G era. In: 2020 IEEE 15th International Conference on Industrial and Information Systems (ICIIS) (2020)
2. Cai, Z., Duan, Z., Li, W.: Exploiting multi-dimensional task diversity in distributed auctions for mobile crowdsensing. *IEEE Trans. Mob. Comput.* **20**, 2576–2591 (2020)
3. Fayaz, M., Mehmood, G., Khan, A., Abbas, S., Fayaz, M., Gwak, J.: Counteracting selfish nodes using reputation based system in mobile ad hoc networks. *Electronics* **11**(2), 185 (2022)
4. Feigenbaum, J., Schapira, M., Shenker, S.: Distributed algorithmic mechanism design. In: *Algorithmic Game Theory*, vol. 14, pp. 363–384. Cambridge University Press, Cambridge (2007)
5. Garcia, M.H.C., et al.: A tutorial on 5G NR V2X communications. *IEEE Commun. Surv. Tutorials* **23**(3), 1972–2026 (2021)
6. Jedari, B., Di Francesco, M.: Auction-based cache trading for scalable videos in multi-provider heterogeneous networks. In: *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pp. 1864–1872. IEEE (2019)
7. Liang, H., Li, H., Zhang, W.: A combinatorial auction resource trading mechanism for Cybertwin based 6G network. *IEEE Internet Things J.* **8**(22), 16349–16358 (2021)
8. Liu, X., Qiu, Q., Lv, L.: An online combinatorial auction based resource allocation and pricing mechanism for network slicing in 5G. In: 2019 IEEE 19th International Conference on Communication Technology (ICCT), pp. 908–913. IEEE (2019)
9. Ning, Z., et al.: Partial computation offloading and adaptive task scheduling for 5G-enabled vehicular networks. *IEEE Trans. Mob. Comput.* **21**(4), 1319–1333 (2022)
10. Peng, X., Ota, K., Dong, M., Zhou, H.: Online resource auction for edge-assistant vehicular network with non-price attributes. *IEEE Trans. Veh. Technol.* **70**, 7127–7137 (2021)
11. Pruhs, K., Nisan, N., Roughgarden, T., Tardos, É., Vazirani, V.V. (eds.): *Algorithmic Game Theory*. Cambridge University Press, Cambridge (2007). ISBN 9780521872829, 776 pp. *Oper. Res. Lett.* **36**(5), 656 (2008)
12. Shneidman, J., Parkes, D.C.: Specification faithfulness in networks with rational nodes. In: *PODC 2004*, pp. 88–97. Newfoundland (2004)
13. Sun, W., Liu, J., Yue, Y., Wang, P.: Joint resource allocation and incentive design for blockchain-based mobile edge computing. *IEEE Trans. Wireless Commun.* **19**, 6050–6064 (2020)

14. Wang, P., Xu, N., Sun, W., Wang, G., Zhang, Y.: Distributed incentives and digital twin for resource allocation in air-assisted internet of vehicles. In: 2021 IEEE Wireless Communications and Networking Conference (WCNC) (2021)
15. Yang, S., et al.: On designing distributed auction mechanisms for wireless spectrum allocation. *IEEE Trans. Mob. Comput.* **18**(9), 2129–2146 (2018)

Information and Coding Theory for Wireless Networks



Communication Optimization in Heterogeneous Edge Networks Using Dynamic Grouping and Gradient Coding

Yingchi Mao^{1,2}, Jun Wu^{2(✉)}, Xiaoming He², Ping Ping^{1,2}, and Jianxin Huang³

¹ Key Laboratory of Water Big Data Technology of Ministry of Water Resources,
Hohai University, Nanjing 211100, China

² School of Computer and Information, Hohai University, Nanjing 211100, China
yingchimao@hhu.edu.cn, 1606010225@hhu.edu.cn

³ Research and Development Department Suma Technology Co., Ltd., Suzhou, China
Huangjx@cancon.com.cn

Abstract. Communication load in heterogeneous edge networks is becoming heavier because of excessive computation and delay caused by straggler dropout, leading to high electricity cost and serious greenhouse gas emissions. To create a green edge environment, we focus on mitigating computation and straggler dropout to improve the communication efficiency during the distributed training. Therefore, we propose a novel scheme named Dynamic Grouping and Heterogeneity-aware Gradient Coding (DGHGC) to speed up average iteration time. The average iteration time is used as a metric reflecting the effect of mitigating computation and straggler dropout. Specifically, DGHGC firstly uses the static grouping to evenly distribute stragglers in each group. After the static grouping, considering the nonuniform distribution of nodes due to straggler dropout during the training process, a dynamic grouping depending on dropout frequency of stragglers is employed. The dynamic grouping tolerates more stragglers by examining the dropout threshold to improve the rationality of the static grouping for stragglers. In addition, DGHGC applies a heterogeneity-aware gradient coding to allocate reasonable data to stragglers based on their computing capacity and encode gradients to prevent stragglers from dropping out. Numerical results demonstrate that the average iteration time of DGHGC can be reduced largely compared to the state-of-art benchmark schemes.

Keywords: Heterogeneous edge networks · Communication efficiency · Dynamic grouping · Gradient coding

1 Introduction

With the increasing computation power of edge devices, *e.g.*, smartphones and IoT sensors [1], training Deep Neural Network (DNN) models on multiple edge devices becomes feasible. The distributed model training in edge networks takes

advantages of the distributed parameter server (PS) architecture, where edge devices work as working nodes and the edge server works as the central PS [2].

One of the main challenges for distributed training in heterogeneous edge networks lies in the heavy communication load caused by excessive computation and straggler dropout. Specifically, some nodes incur delay in computation or communication, which are called stragglers. The PS waits for stragglers to submit their local gradients, and even abandons stragglers that cannot return outcomes with a reasonable deadline, which are called straggler dropout. This largely increases the communication cost and reduces the accuracy of the trained model [3]. In the meantime, the attributes of heterogeneous nodes, *i.e.*, the computing capacity [4], memory size [5], and communication mode [6], further exacerbate straggler dropout, resulting in the inefficient communication. Therefore, it is vital to design an efficient solution of straggler dropout for green communication and computing.

The straggler dropout has been widely studied in distributed computing [7]. From the perspective of parallelism mechanism, the typical Time Asynchronous Parallel (TAP) [8] and Staleness Synchronous Parallel (SSP) [9] were proposed to mitigate the negative impact of stragglers on the communication efficiency. Based on the SSP algorithm, Dynamic Stochastic Gradient Descent (DynSGD) [10] adjusted the learning rate dynamically according to the delay of stragglers in completing the gradient computation. Considering the above parallelism schemes alleviated the model accuracy, Raviv *et al.* [11] proposed gradient Coding (GC) to tolerate stragglers by encoding gradients. However, GC is only applicable in homogeneous training environments, which does not work in heterogeneous edge networks. Given the heterogeneity of edge devices, Wang *et al.* [12] proposed Heterogeneity-aware Gradient Coding (HGC) to allocate reasonable data to nodes depending on their computing capacity, which tolerates a predetermined number of stragglers to prevent them from dropping out.

To achieve the expected communication efficiency, GC and HGC are designed for the static grouping of heterogeneous edge devices. However, they fail to consider the fact that stragglers may drop out after the static grouping. To address the shortcomings of the static grouping for stragglers, Buyukates *et al.* [13] proposed Dynamic Clustering with Gradient Coding (DCGC), with the goal of dynamically adjusting the number of stragglers in each cluster based on the straggler dropout in the previous iteration, making stragglers uniformly dispersed into each cluster to the maximum extent. However, DCGC ignores the heterogeneity of edge devices and fails to fully utilize the computation power of nodes for the data allocation.

Considering the strong heterogeneity of nodes and inappropriate groupings for stragglers that leads to straggler dropout in edge networks, we adopt a novel grouping for stragglers, which firstly employs the static grouping and then utilizes the dynamic grouping to improve the rationality of the static grouping for stragglers. Besides, a heterogeneity-aware gradient coding is applied in both grouping phrases. Therefore, we propose a scheme termed Dynamic Grouping and Heterogeneity-aware Gradient Coding (DGHGC). DGHGC mitigates communication load by solving heavy computation and straggler dropout in heterogeneous edge networks.

The main contributions of this paper include,

- Two static groupings based on greedy algorithm and Karmarkar-Karp (KK) algorithm [14] are used to evenly distribute stragglers in each group, mitigating inherent heterogeneity gaps in edge networks. Given the fact that the static grouping ignores dynamic straggler dropout during the actual training, a dynamic grouping based on the frequency of straggler dropout is proposed.
- Considering the heterogeneity of nodes in dynamic grouping, we propose a novel scheme termed DGHGC. Specifically, DGHGC allocates data of reasonable size to stragglers and encodes uploaded gradients depending on their computing capacity, which tolerates a predetermined number of stragglers to prevent them from dropping out and mitigates delay caused by straggler dropout.
- Evaluation of baseline comparison demonstrates that DGHGC reduces the average iteration time by about 2.3 times, 1.53 times, 1.58 times and 1.45 times, respectively, compared with GC, HGC, GCC, and DCGC.

The rest of this paper is organized as follows. Section 2 presents the system model. In Sect. 3, DGHGC is discussed in detail. Then, DGHGC is compared with common gradient coding approaches in Sect. 4. At last, conclusions are drawn in Sect. 5.

2 System Model and Problem Formulation

Suppose that the distributed system in the edge environment has m working nodes $\{W_1, W_2, \dots, W_m\}$, with the computation power $\mathcal{C} : \{c_1, c_2, \dots, c_m\}$. Considering the heterogeneity of nodes, the PS divides these nodes into k groups. The data set \mathcal{D} is also evenly divided into k non-overlapping copies, expressed as $\{\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_k\}$. Each group has n working nodes ($\lceil n = \frac{m}{k} \rceil$, n is an integer), and the group-based approach is used for m nodes. The data \mathcal{D}_i of each group is divided into d_i copies that do not overlap with each other, represented by $\mathcal{D}_i : \{\mathcal{D}_i^1, \mathcal{D}_i^2, \dots, \mathcal{D}_i^{d_i}\}$ and d_i is calculated as,

$$d_i = c_i^{total} = \sum_{j=1}^n c_i^j, \quad (1)$$

where c_i^{total} represents the total capacity of all working nodes in group i , and c_i^j stands for the computational capacity of the node j in i . To tolerate s stragglers, each data set in the group has to be redundantly stored in r copies (r refers to the redundancy factor and $r = s + 1$), and there are $d_i(s + 1)$ copies of data in total. The number of data copies $amount_i^j$ allocated by j in i is calculated as,

$$amount_i^j = d_i \cdot (s + 1) \cdot \frac{c_i^j}{c_i^{total}} = (s + 1) \cdot c_i^j. \quad (2)$$

The overall time for each iteration is expressed as,

$$T^{(t)} = \max_{i \in \{1, 2, \dots, k\}} \{T_i^{(t)}\} \quad (3)$$

where $T^{(t)}$ denotes the overall iteration time after the number of training iteration t , k represents the number of groups and $T_i^{(t)}$ stands for the iteration time needed for the training of i after t training iteration. $T_i^{(t)}$ depends on the slowest working node within the group as,

$$T_i^{(t)} = \max_{j \in \{1, 2, \dots, n\}} \{Total_T_i^{j(t)}\}, \quad (4)$$

where $Total_T_i^{j(t)}$ represents t of j in i during the iteration training, and n refers to the number of nodes in each group. $Total_T_i^{j(t)}$ consists of two parts, *i.e.*, computation time $Comp_T_i^{j(t)}$ and communication time $Comm_T_i^{j(t)}$. That is,

$$Total_T_i^{j(t)} = Comp_T_i^{j(t)} + Comm_T_i^{j(t)}. \quad (5)$$

- Computation time: It depends on the amount of data allocated and the computation capacity of nodes. The computation time of each working node is expressed as,

$$Comp_T_i^{j(t)} = \frac{\|\mathcal{I}_i^j\|}{c_i^j}, \quad (6)$$

where $\|\mathcal{I}_i^j\|$ and c_i^j represent the data volume and the computation capacity of j in i , respectively.

- Communication time: It is determined by the network condition and the amount of the transferred data.

To sum up, the total time for each iteration of the training is expressed as,

$$T^{(t)} = \max_{i \in [1, k]} \{ \max_{j \in [1, n]} \left\{ \frac{\|\mathcal{I}_i^j\|}{c_i^j} + Comm_T_i^{j(t)} \right\} \}. \quad (7)$$

To improve the communication efficiency, the overall optimization objective is set to minimize the total time of each iteration training as,

$$\arg \min T^{(t)}. \quad (8)$$

Due to the reduction of the model accuracy caused by straggler dropout, tolerating a maximum number of stragglers to take full utilization of their computing resources becomes necessary. However, the more stragglers, more significantly the computing time of the system will increase. Trading off the model accuracy and total system time influenced by the number of stragglers is worth considering. DGHGC makes each node possible to have the similar completion time so that the consistent straggler dropout incurred by heterogeneity could be eliminated.

3 Dynamic Grouping and Heterogeneity-Aware Gradient Coding

DGHGC is proposed for the overall optimization objective. The implementation process of DGHGC, as shown in Fig. 1, is divided into two steps. The first step

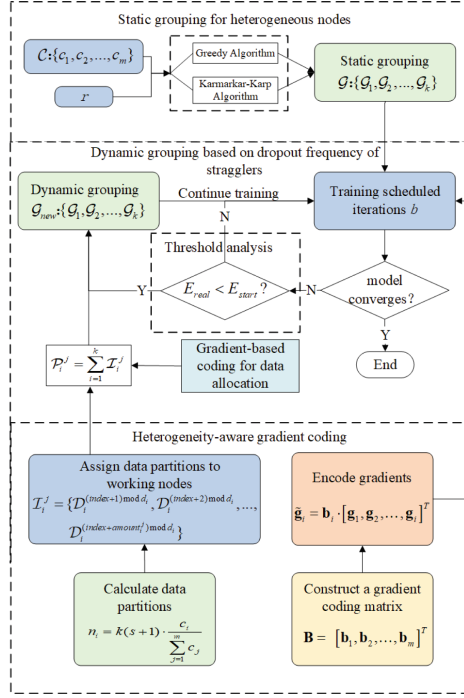


Fig. 1. The implementation process of DGHGC

refers to grouping heterogeneous nodes, including static grouping and dynamic grouping based on the dropout frequency of stragglers. That is, stragglers are evenly distributed in each group and more stragglers are tolerated to improve the communication efficiency. For the second step, the heterogeneity-aware gradient coding is applied within each group to make full use of the computation power of heterogeneous working nodes and encode gradients to prevent them from dropping out.

3.1 Static Grouping and Dynamic Grouping

In the static grouping, all nodes are divided into k groups. Each group tolerates $r - 1$ stragglers. k is limited by the following factors,

- To tolerate more stragglers than HGC, $k > r - 1$.
- The number of data copies should satisfy $amount_i^j \leq d_i$ to obtain $r \cdot \frac{c_i^j}{c_i^{total}} \leq 1$. To satisfy the case that all nodes have the most computational power, $c_i^{total} \geq r \cdot c_{max}$ is required, where c_{max} represents the node with the most computational power, and the number of k has to satisfy $k \leq \frac{c^{total}}{r \cdot c_{max}}$.

In summary, the number of groups k should satisfy the following,

$$k \in (r - 1, \frac{c^{total}}{r \cdot c_{max}}]. \quad (9)$$

We design a static grouping approach based on the greedy algorithm and Karmarkar-Karp (KK) [14] to obtain the optimal grouping for stragglers. The specific process of the static grouping is as follows,

- 1) Input the set $\mathcal{C} : \{c_1, c_2, \dots, c_m\}$ of the computation power of m nodes and the redundancy factor r . Then, set $k = k_{max}$ using (9). At last, sort set \mathcal{C} in decreasing order.
- 2) Judge differences in set \mathcal{C} [14]. If differences are small, the static grouping based on the greedy algorithm are employed. Specifically, initialize k empty sets $\{\mathcal{G}_1, \mathcal{G}_2, \dots, \mathcal{G}_k\}$ firstly. Then, consider one element in set \mathcal{C} at a time, and place it in the subset with the smallest sum so far. In the case of subsets with an equal sum, choose any subset until set \mathcal{C} is empty. While differences are large, we use Karmarkar-Karp (KK) to realize the static grouping. That is, a list of k -tuples is created for each integer in set \mathcal{C} . The first integer of this tuple is the value in set \mathcal{C} and the rest integers are set to 0. Combine the first two tuples in the list (the two tuples with the largest integers). Given the tuples $A = (a_1, a_2, \dots, a_n)$ and $B = (b_1, b_2, \dots, b_n)$, both sorted in decreasing order, A is combined with B to form a new tuple C as,

$$C = (a_1 + b_n, a_2 + b_{n-1}, \dots, a_n + b_1). \quad (10)$$

After the combination of tuples A and B , the KK algorithm keeps track of the relative differences between subsets only, and normalizes set C by subtracting its minimum value. This merging process continues until only one tuple remains and the algorithm ends.

- 3) Determine whether the sum of each subset satisfies the condition $c_i^{total} \geq r \cdot c_{max}$. If yes, output the group. Otherwise, make the groups' amount $k = k - 1$, and re-execute the static grouping approach.

After the static grouping, the dynamic grouping is employed to improve the rationality of groupings for stragglers. Dynamic grouping based on the dropout frequency of stragglers includes two key steps, *i.e.*, threshold analysis and dynamic adjustment.

Threshold Analysis. Firstly, the number of iterations b for the static grouping training is predetermined. Then, it is determined whether to dynamically adjust stragglers groups depending on the actual expectation after b training times. The specific steps are as follows,

- Calculate the expectation E_{start} of the tolerable straggler number based on the assumption that k nodes has an equal probability to become stragglers in each iteration.

- Perform b iterations, and calculate the actual expectation E_{real} of the number of tolerable stragglers by the frequency of becoming a straggler of each node. Determine whether the actual expectation E_{real} reaches the threshold, *i.e.*, the static expectation E_{start} . If not, that is $E_{real} < E_{start}$, the effect of the static grouping is poor and has to be dynamically adjusted until $E_{start} = E_{real}$. Otherwise, no adjustment is needed.

Dynamic Adjustment. In general, dynamic adjustment includes three steps. Firstly, heterogeneity-aware gradient coding is applied within each group to assign data \mathcal{I}_i^j to nodes. That is,

$$\mathcal{I}_i^j = \{\mathcal{D}_i^{(index+1) \bmod d_i}, \mathcal{D}_i^{(index+2) \bmod d_i}, \dots, \mathcal{D}_i^{(index+amount_i^j) \bmod d_i}\}, \quad (11)$$

where $index$ denotes the index of nodes in the group, d_i means dividing data \mathcal{D}_i into d_i copies equally, and $amount_i^j$ indicates the number of data copies by j in i . Then, to make intergroup nodes replaceable, nodes with comparable computational power between groups need to have data from each other. That is, allocate alternative data $\mathcal{P}_i^j = \sum_{i=1}^k \mathcal{I}_i^j$ to the nodes, where k represents the number of groups. Secondly, estimate k nodes with high probability to become stragglers based on previous iterations. Thirdly, set the group priority of stragglers according to their number and assign $n - r + 1$ non-stragglers to each group based on the priority, which tolerates $r - 1$ stragglers. If non-stragglers are not enough, offer the priority to non-stragglers and then assign the alternative nodes of stragglers based on data allocation \mathcal{P}_i^j until all groups are allocated.

3.2 Dynamic Grouping and Heterogeneity-Aware Gradient Coding

Besides realizing the node grouping, DGHGC also fully exploits computation resources of heterogeneous nodes for data allocation and encodes gradients that adapt to the computation power of stragglers, reducing delay caused by straggler dropout. For data allocation, each data partition \mathcal{I}_i^j has to be assigned with at least $s + 1$ nodes to tolerate s stragglers by using (11). For the coding phrase, to prevent stragglers from dropping out, gradients coding matrix B is employed in each group as,

$$\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_m]^T, \quad (12)$$

where $b_j = 1$ if $\mathcal{D}_i^j \in \mathcal{I}_i^j$, else $b_j = 0$. By constructing coding matrix B , we encode gradients that adapt to the computation power of stragglers as,

$$\tilde{\mathbf{g}}_i = \mathbf{b}_i \cdot [\mathbf{g}_1, \mathbf{g}_2, \dots, \mathbf{g}_i]^T, \quad (13)$$

where g_i means the gradient that aggregating the partial gradient of each node in Group i , and $\tilde{\mathbf{g}}_i$ represents encoded gradients by B for the global gradient aggregation of the server. At last, we describe the workflow of DGHGC in Algorithm 1.

3.3 DGHGC Complexity

The complexity of greedy algorithm and Karmarkar-Karp algorithm are both $O(2^m)$ [14], where m is the number of integers in the input set \mathcal{C} . Considering the dynamic grouping, the complexity is $O(s)$ with s stragglers. Given the gradient code, the complexity is $O\left(\frac{1}{\sqrt{kP}}\right)$ with the group number k and iteration time P [12]. Thus, the overall complexity of DGHGC is $O\left(2^m + s + \frac{1}{\sqrt{kP}}\right)$, which is slightly higher than HGC.

Algorithm 1 Dynamic Grouping and Heterogeneity-aware Gradient Coding

Input: k : number of groups, r : redundancy number, $\mathcal{C} : \{c_1, c_2, \dots, c_m\}$: computation power of nodes, b : training time for the static grouping, \mathcal{I}_i^j : data partition for the node j in the group i , n : number of nodes in each group, m : total number of nodes, s : total number of stragglers

Output: $\tilde{\mathbf{g}}_i$: encoded gradient

- 1: **Initialize the static grouping:**
- 2: Set $k = k_{max}$ by using (9)
- 3: Judge differences in Set $\mathcal{C} : \{c_1, c_2, \dots, c_m\}$
- 4: **if** differences are small **then**
- 5: static grouping based on the greedy algorithm
- 6: **else**
- 7: static grouping based on the Karmarkar-Karp algorithm
- 8: **end if**
- 9: **Conduct the dynamic grouping:**
- 10: Perform b distributed training
- 11: Calculate E_{start} by the probability to become a straggler for each node
- 12: Calculate E_{real} by the frequency to become a straggler for each node
- 13: **if** $E_{real} < E_{start}$ **then**
- 14: **repeat**
- 15: Allocate data \mathcal{I}_i^j to all nodes by using (11)
- 16: Assign $n - r + 1$ non-stragglers to each group to tolerate $r - 1$ stragglers
- 17: **until** $E_{start} = E_{real}$
- 18: **else**
- 19: **break**
- 20: **end if**
- 21: Encode gradients based on Heterogeneity-aware Gradient Coding by using (12-13)
- 22: **return** $\tilde{\mathbf{g}}_i$

4 Experiments

4.1 Experiment Settings

The experiments are carried out with two DELL PowerEdge R740 servers and 20 nodes with different CPU processing power. Each server is equipped with

two 28-core Intel Xeon Platinum 8180M CPUs and one NVIDIA GeForce RTX 3090 GPU. The memory capacity of each server is 93 GB. 5, 3, 3, 4, 5 nodes with CPU main frequencies of 600 MHz, 1.2 GHz, 1.8 GHz, 2.4 GHz and 3 GHz are selected. The memory capacity of each CPU node is 32 GB. The software platform of our experiments is PyTorch, which is a deep learning experimental platform that provides a high degree of flexibility and efficiency.

Actually, complex network environments, unstable transmission rates and heterogeneous hardware properties between edge nodes definitely affect the algorithm performance. We only consider the most influential factor that causes straggler dropout, that is the CPU processing power.

In addition, other experimental settings draw lessons from [12]. The number of iterations I is set to 500, the learning rate α is 0.01, and the redundancy factor r is set to 2. Besides, the data transmission rate between nodes and the server is 3–5 MB/s.

4.2 Analysis of Results

Table 1. DGHGC at different b values for 20 nodes with 5 groups

b	CNN(MNIST)		CNN(CIFAR-10)		LeNet5(CIFAR-10)	
	$T(s)$	$Acc\%$	$T(s)$	$Acc\%$	$T(s)$	$Accuracy\%$
100	0.90	93.6	0.92	60.9	0.96	62.6
50	0.56	93.6	0.54	61.2	0.59	62.5
30	0.68	93.8	0.64	60.8	0.70	62.8
20	0.75	93.4	0.72	61.0	0.81	62.4
10	0.87	93.2	0.83	61.1	0.89	62.5

Analysis of Parameter Values. The parameter value of b in DGHGC indicates that one dynamic adjustment is executed after b rounds of iterations. The value of b affects the number of dynamic groupings and the grouping effect of DGHGC. Five different values of b are selected for the comparative experiments with 20 nodes. The experimental results are shown in Table 1, and each value is averaged after 10 independent experiments.

It can be observed from Table 1 that the value of b can hardly affect the model accuracy. When $b = 100$, the communication cost with DGHGC is the least efficient. This is due to the fact that the number of samples is large enough, yet the number of groupings is small and close to the static grouping, leading to the worst communication efficiency, while the best communication efficiency is achieved when $b = 50$. The average iteration time of the distributed training increases with the decrease of b . This is because when b decreases gradually, the number of dynamic groupings becomes large, which leads to a certain time delay. Secondly, due to the small number of samples, the results of stragglers dynamic grouping are not ideal initially, which reduces the communication efficiency. The value of b is determined as 50 by experiments, and applied into the comparison experiments below.

Analysis of Communication Efficiency. Figure 2 shows the experimental results of the average iteration time based on five approaches, *i.e.*, GC, HGC, GCC, DCGC and DGHGC for training the MNIST dataset and CIFAR-10 dataset on the CNN model and the CIFAR-10 dataset on the LeNet5 model under 20 nodes.

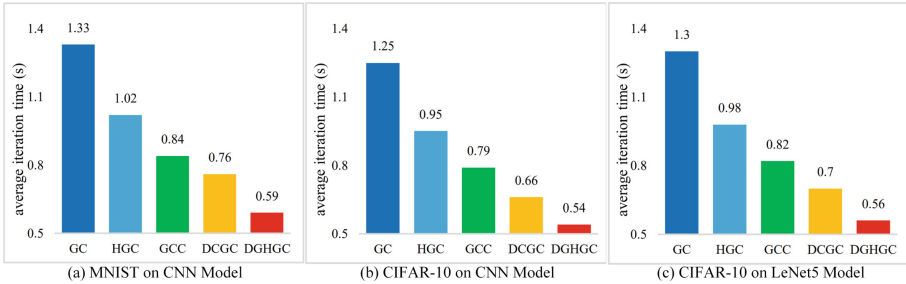


Fig. 2. Average iteration time for 20 nodes with 5 groups

It can be observed from Fig. 2 that HGC, GCC, DCGC and DGHGC significantly outperform the traditional algorithm GC in terms of the communication efficiency. DCGC combines the idea of dynamic grouping based on GCC so as to tolerate more stragglers, and make its communication efficiency performance better than that of GCC. The communication efficiency of GCC and DCGC is better than that of HGC in the distributed training. In all experiments results, DGHGC achieves the best communication efficiency. The reason is that DGHGC not only considers the node heterogeneity, but also combines the dynamic grouping to distribute stragglers in each group to the maximum extent. Moreover, DGHGC does not perform the dynamic grouping in each iteration, but sets a parameter to reduce the number of adjustments, thus reducing the time consumption, compared with DCGC. The average speed of DGHGC is about 2.3 times than that of the traditional algorithm GC, about 1.53 times, 1.58 times, 1.45 times than those of the state-of-art algorithms HGC, GCC, and DCGC.

The following experiment studies the effects of different groups k on DGHGC, and the results are shown in Fig. 3. According to Equation (9), Group A is divided into 5 groups ($k = 5$) at the most. It can be observed that the average iteration time becomes shorter with the increase of k . The larger k becomes, the more obvious the greater number of stragglers tolerated in each iteration under DGHGC, resulting in shorter iteration time and an improvement of communication efficiency.

Analysis of Training Accuracy. Table 2 shows the results of training accuracy obtained from MNIST dataset and CIFAR-10 dataset on CNN model, and from CIFAR-10 dataset on LeNet5 model under the five approaches, *i.e.*, GC, HGC, GCC, DCGC and DGHGC with 20 nodes.

As it can be observed from Table 2, there is almost no difference on training accuracy under five approaches. The reason is that all these approaches utilize the gradient encoding to store the data redundantly without changing gradients. Therefore, DGHGC does not suffer from the decrease of the model training accuracy.

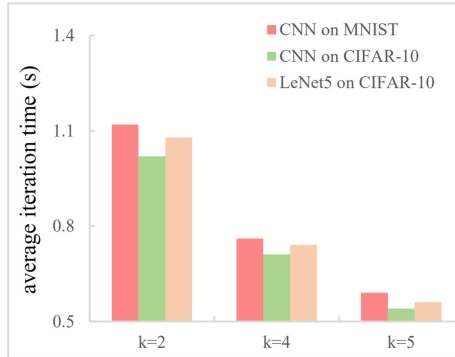


Fig. 3. Average iteration time of DGHGC with different number of k

Table 2. Training accuracy

APPROACH	ACCURACY(%)		
	CNN (MNIST)	CNN (CIFAR-10)	LeNet5 (CIFAR-10)
GC	93.2	61.5	62.7
HGC	93.3	61.1	62.6
GCC	93.6	61.1	62.4
DCGC	93.4	61.4	62.4
DGHGC	93.6	61.2	62.5
AVERAGE	93.4	61.2	62.5

5 Conclusion

In the edge environment, the differences of computation power of terminal equipment lead to the severe stragglers dropout in the distributed training, which reduces the communication efficiency and causes plenty of electricity cost and greenhouse gas emissions. To achieve green communication and computing, a Dynamic Grouping and Heterogeneity-aware Gradient Coding (DGHGC) scheme is proposed. DGHGC employs the dynamic grouping and gradient coding to solve straggler dropout in heterogeneous edge networks. From the results, DGHGC can effectively improve the communication efficiency compared to the benchmark schemes.

Acknowledgements. This work is supported by The Key Research and Development Program of China (No. 2022YFC3005400), Key Research and Development Program of China, Yunnan Province (No. 202203AA080009), The National Natural Science Foundation of China (No. 61902110), Postgraduate Research & Practice Innovation Program of Jiangsu Province (No. KYCX22_0610), The Fundamental Research Funds for The Central Universities (No. B210203024), Transformation Program of Scientific and Technological Achievements of Jiangsu Province (No. BA2021002), the Key Research and Development Program of China, Jiangsu Province (No. BE2020729) and the Key Technology Project of China Huaneng Group (Grant No. HNKJ20-H46).

References

1. Daghero, F., Pagliari, D.J., Poncino, M.: Energy-efficient deep learning inference on edge devices. *Adv. Comput.* **122**, 247–301 (2021)
2. Zhou, Q., et al.: Falcon: addressing stragglers in heterogeneous parameter server via multiple parallelism. *IEEE Trans. Comput.* **70**(1), 139–155 (2020)
3. Li, W., Liu, D., Chen, K., Li, K., Qi, H.: Hone: mitigating stragglers in distributed stream processing with tuple scheduling. *IEEE Trans. Parallel Distrib. Syst.* **32**(8), 2021–2034 (2021)
4. Zhou, Q., et al.: Petrel: heterogeneity-aware distributed deep learning via hybrid synchronization. *IEEE Trans. Parallel Distrib. Syst.* **32**(5), 1030–1043 (2020)
5. Zhou, A., et al.: Brief industry paper: optimizing memory efficiency of graph neural networks on edge computing platforms. In: 27th IEEE Real-Time and Embedded Technology and Applications Symposium, RTAS, pp. 445–448 (2021)
6. Kai, J., Zhou, H., Chen, X., Zhang, H.: Mobile edge computing for ultra-reliable and low-latency communications. *IEEE Commun. Stand. Mag.* **5**(2), 68–75 (2021)
7. Lu, H., Wang, K.: Distributed machine learning based mitigating straggler in big data environment. In: IEEE International Conference on Communications, ICC (2021)
8. Yuan, G., et al.: Improving DNN fault tolerance using weight pruning and differential crossbar mapping for ReRAM-based edge AI. In: IEEE International Symposium on Quality Electronic Design, pp. 135–141. ISQED (2021)
9. Zhou, Q., et al.: Petrel: community-aware synchronous parallel for heterogeneous parameter server. In: 40th IEEE International Conference on Distributed Computing Systems, ICDCS, pp. 1183–1184 (2020)
10. Miao, X., et al.: Heterogeneity-aware distributed machine learning training via partial reduce. In: SIGMOD 21th International Conference on Management of Data, SIGMOD, pp. 2262–2270 (2021)
11. Raviv, N., Tamo, I., Tandon, R., Dimakis, A.G.: Gradient coding from cyclic MDS codes and expander graphs. *IEEE Trans. Inf. Theory* **66**(12), 7475–7489 (2020)
12. Wang, H., Guo, S., Tang, B., Li, R., Li, C.: Heterogeneity-aware gradient coding for straggler tolerance. In: 2019 IEEE 39th International Conference on Distributed Computing Systems, ICDCS, pp. 555–564 (2019)
13. Buyukates, B., Ozfatura, E., Ulukus, S., Gunduz, D.: Gradient coding with dynamic clustering for straggler mitigation. In: IEEE International Conference on Communications, ICC (2021)
14. Schreiber, E.L., Korf, R.E., Moffitt, M.D.: Optimal multi-way number partitioning. In: Dissertations and Theses - Gradworks vol. 65, no. (4), pp. 24:1–24:61 (2018)



Design on Rateless LDPC Codes for Reliable WiFi Backscatter Communications

Sicong Xu¹, Xin He^{1,3}(✉), Fan Wu¹, Guiping Lin¹, and Panlong Yang^{2,3}

¹ School of Computer and Information, Anhui Normal University, Wuhu, China
xin.he@ahnu.edu.cn

² School of Computer Science and Technology, University of Science and Technology
of China, Hefei, China

³ Deqing Alpha Innovation Institute, Deqing, China

Abstract. This paper designs a rateless low density parity check (LDPC) code for the information transmission of the WiFi backscatter communications. Since WiFi has the characteristics of burst data packages and low anti-jamming capability, the reliability becomes a problem. Therefore, the encoding is significantly crucial when using WiFi signals as the excitation in backscatter communications. Rateless LDPC code can be applied to not only solve these two shortcomings, but also adjust the link state and the bit rate without knowing the channel state information. It ensures that transmission resources are not wasted and the computational resources are saved. We conduct simulation experiments and the obtained results show that the rateless LDPC still performs well under the restriction of the number of retransmissions. Furthermore, the proposed scheme works against the intermittent nature of WiFi excitation signals.

Keywords: WiFi backscatter · LDPC code · Rateless code · Iterative decoding

1 Introduction

In recent years, with the increasing number of Internet of things (IoT) devices, the backscatter communication [1] is considered as a promising solution to tackle the challenges of energy supply and consumption. Among these backscattering systems, WiFi-based backscatter communication is ubiquitous. However, due to the low transmission power of backscatter signal excited by the WiFi signal, the

This work is in part funded by NSFC under Grant No. 62072004, and in part supported by the University Synergy Innovation Program of Anhui Province under Grant No. GXXT-2019-024.

receiver usually receives an extremely low power signal. Moreover, the signal power is exponentially decayed as the operation distance increasing. It is then a hard task for the receiver to detect and decode the backscatter signal. In addition, WiFi packets are bursty [2, 3] in nature, so that the data to be backscattered may not have an excitation signal. Currently, there are two major approaches to overcome these difficulties to improve the system performance; one is to use more complex encoding and decoding algorithms, and the other is to control data transmission by analyzing and predicting network conditions.

Complicated encoding and decoding algorithms are used to find and correct possible errors by increasing the number of transmitting bits and the correlation between each bit. Analyzing and predicting network conditions can find out those low-power tags are limited by their capabilities to adjust the backscatter transmission strategy by themselves, while when occurs the interval between data packets, that can be used to adjust reflective transmission strategies of backscatter tags. GuardRider [4] designed an optimization algorithm of Reed-Solomon codes to fit the statistical changes of the WiFi traffic and adjust backscatter transmission policy. Reference [5] proposed a 2×2 space-time code (STC) for backscatter communication. Reference [6] investigated the bit error rate (BER) performance of an ambient backscatter communication (AmBC) system that uses LDPC-coded radio frequency (RF) source signals. Manchester coding and differential Manchester coding are adopted at the information tag of an AmBC system, and the corresponding semicoherent Manchester (SeCoMC) and non-coherent Manchester (NoCoMC) detectors are developed in [7].

However, adjusting the transmission schemes, e.g., different coding rates, requires the channel state information at the tag. Due to the scarce computing resources of the tag, it is extremely hard to estimate of the channel by the tag. The tag is then impossible to adjust coding and/or link rates to improve the system performance. Therefore, the WiFi backscatter communication usually encounter a low throughput in real deployment. To this end, we propose a rateless low density parity check (LDPC) code scheme to tackle the problem. It is worthwhile to mention that the rateless LDPC code gain both the benefits from the LDPC codes and rateless codes. In particular, LDPC codes are proven to be Shannon capacity achieving codes if the coding parameters are well designed. On the other hand, the rateless codes do not require the channel state information. The tag can continually transmit the coded packets until it receives the positive acknowledgement from the receiver. As a whole, the tag can operate using a low power consumption and the system performance is enhanced, by applying the designed rateless LDPC codes.

The major contributions of this work are summarized as follows.

- We propose LDPC code combined with rateless method through the special check matrix. The check matrix consists of two parts, one of which is derived from the index matrix. Rateless method is supported by changing the exponent of prime number in index matrix, which allows us to create countless check matrix.

- Based on characteristics of rateless code, backscatter system is endowed with the ability to adapt to the channel states. If the channel state is poor, the tag automatically increases the number of coded packets in order to achieve reliable transmissions. In the opposite case, the tag rises the bit rate to ensure that transmission resources are not wasted.
- On the premise that the code length is 1310, BER reaches 10^{-6} orders of magnitude when SNR is about 2.5 dB, while rateless LDPC code gains 0.5 dB in SNR. As the number of coded packets increases, the required SNR gets smaller. However, when the excitation interval ratio is 0.1, BER reach at orders of 10^{-5} if SNR= 5 dB.

2 Preliminaries

In this section, we give the necessary preliminary knowledge about this work, including the backscatter communications, LDPC code and the rateless code.

2.1 Backscatter Communication Based on WiFi

Backscatter communication is now gradually applied to the data transmission of IoT terminal due to the low energy consumption [8]. It usually consists of three parts: excitation source, backscatter tag and receiver. The backscatter tag receives the RF signal from the excitation source, modulates it by adjusting the load impedance, and transmits it to the receiver. The tag itself does not generate RF signal but uses RF signal from the excitation source to transmit data, resulting in the ultra-low power consumption of backscatter communication.

In daily life, TV signal [9], Bluetooth signal [10], FM signal [11], WiFi signal, et al. can be used as RF signal. Among them, the WiFi signal sent by the wireless access point (AP) as the excitation source can be called WiFi backscattering communication. WiFi signal is ubiquitous in real life, and the AP is easy to deploy. There is no need to add additional devices other than the backscatter tags. However, there are also some problems, such as the severe signal interference during the long-distance transmission and gaps between WiFi packets, also known as WiFi interruption, which cause information lost of the receiver. Therefore, we propose rateless LDPC code for reliable transmission.

2.2 LDPC Code and Rateless Code

LDPC Code. LDPC code is a linear grouping code with a sparse check matrix which is invented by Prof. Gallager of MIT in 1963 [12,13]. It not only has a near Shannon limit performance [14], but also has a low decoding complexity and a flexible structure. Characterized by a small number of non-zero elements in its check matrix, it maps a sequence of information into a coded sequence by a generator matrix \mathbf{G} , which corresponds to a check matrix \mathbf{H} .

LDPC codes are divided into random LDPC codes and quasi-cyclic LDPC codes (QC-LDPC) in coding categories. Although random LDPC performs better, the irregular distribution of 1 in their check matrices requires that all row

vectors of the generated and check matrices must be stored during actual deployment. Thus, very large scale integration (VLSI) encoding and decoding of random LDPC codes are difficult to implement. However, QC-LDPC can solve this problem since its characteristic of quasi-cyclic [15]. There are many methods to generate QC-LDPC check matrix, among which the method of using index matrices is the most common and easiest one.

The form of the index matrix \mathbf{P} is defined in (1), as

$$\mathbf{P} = \begin{pmatrix} 1 & a & \dots & a^{k-1} \\ b & ab & \dots & a^{k-1}b \\ \vdots & \vdots & & \vdots \\ b^{j-1} & ab^{j-1} & \dots & a^{k-1}b^{j-1} \end{pmatrix}, \quad (1)$$

where the size of the matrix \mathbf{P} is $j \times k$, and $P_{s,t} = a^t b^s$ denotes the element in s -th row and t -th column, with prime numbers a and b . The (s, t) cyclic submatrix in the check matrix is generated by cyclic shifting the elements of the identity matrix \mathbf{I} . The cyclic shift of \mathbf{I} in scale of $q \times q$ needs an pan right for each row. As shown in (2), \mathbf{I}_x represents the cyclic shift of x bits to the right in each row of the identity matrix.

$$\mathbf{H} = \begin{pmatrix} \mathbf{I}_1 & \mathbf{I}_a & \dots & \mathbf{I}_{a^{k-1}} \\ \mathbf{I}_b & \mathbf{I}_{ab} & \dots & \mathbf{I}_{a^{k-1}b} \\ \vdots & \vdots & & \vdots \\ \mathbf{I}_{b^{j-1}} & \mathbf{I}_{ab^{j-1}} & \dots & \mathbf{I}_{a^{k-1}b^{j-1}} \end{pmatrix} \quad (2)$$

That is the method of generating QC-LDPC check matrix. With the check matrix, we can easily calculate the generator matrix by the regular operation. In coding theory, if \mathbf{H} can be expressed as $[\mathbf{I}|\mathbf{Q}]$, where \mathbf{I} is an identity matrix, \mathbf{G} can be expressed as $[\mathbf{P}^T|\mathbf{I}]$, where \mathbf{T} means transpose. Then the information is encoded by the generator matrix \mathbf{G} for further processing, e.g., modulation.

In receiver, the data needs to be decoded. The general decoding methods are divided into hard decoding and soft decoding. For hard decoding algorithm, if a bit does not satisfy the maximum number of check equations, it is most likely to be an error bit so that flip it. The hard decoding algorithm discards the reliability information of each bit, and makes a hard judgment on the code word only. Thus, hard decoding has the simplest theory, easiest implementation, but the worst performance. When two consecutive iterations of the flip function determine that the same bit is the most error-prone bit, the algorithm will fall into an infinite loop, which greatly reduces the performance of decoding. In practice, we usually use soft decoding algorithm, which is also known as the Belief propagation (BP) algorithm.

The BP algorithm is an iterative decoding algorithm based on Tanner graph. The Tanner graph represents a check matrix of LDPC codes. Tanner graph contains two types of vertices: N bit vertices (called bit nodes), corresponding to each column of the check matrix, and M check equation vertices (called check nodes), corresponding to each row of the check matrix. Each row of the check

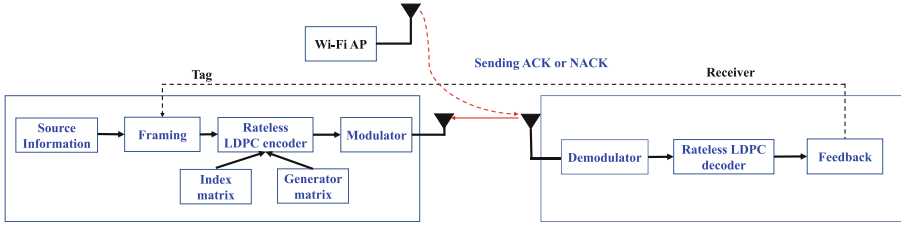


Fig. 1. The main components of the tag and the receiver of using rateless LDPC codes for WiFi backscatter system.

matrix represents a check equation, and each column represents a bit. If there is a bit in the corresponding check equation, the bit nodes and check nodes involved are connected by a line, so the number of lines in Tanner graph is the same as in check matrix.

In the iteration process, messages are passed back and forth through the edges of the Tanner graph in the variable nodes and check nodes. After several iterations, it tends to be converged, and the best decision is made accordingly. Those messages are posterior probabilities, and the initial value of the messages are set to the prior probability of the information bits.

The complexity of the iterative BP algorithm is linearly proportional to the length of the code. Parallel implementation in hardware can greatly improve the decoding speed. If the decoding succeeds in the iteration process, it ends immediately, which can effectively reduce the number of iterations. BP algorithm is widely accepted because it can achieve close channel capacity performance in AWGN environment in both theory and practice.

Rateless Code. The major difference between rateless code [16] and traditional fixed-rate encoding is that it does not set a fixed rate at the sender. The sender of rateless code can generate and send countless encoding packages in some way based on the incoming data packets. The receiver can receive the encoding packages and try to decode them. If decoding fails, the receiver will receive more encoding packages and continue to decode. The receiver can repeat this process until decoding succeeds. Once the decoding succeeds, the receiver only needs to send a simple feedback signal to inform the sender, then the sender will stop sending, thus the transmission process is completed entirely. In this case, the actual transmitted bit rate depends on the number of encoding packets which are actually sent, while the actual number of encoding packets depends on the channel conditions at that time.

Normally, the sending rate should be adjusted according to the channel state to ensure the communication quality. However, the rate of the error-correcting code usually is not so flexible. This is the reason why we use rateless code, which enables reliable delivery, and adapting link rates to ensure that transmission resources are not wasted.

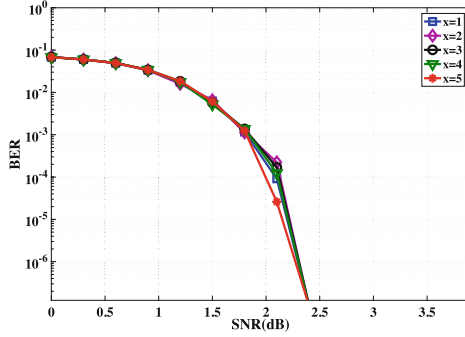


Fig. 2. The BER performance of various initial exponential values.

3 System Design

Figure 1 illustrates an overview of our designed system. Source information collected by tags is encoded by rateless LDPC code after framing. Index matrix and generator matrix are used during encoding process. Encoded frames are modulated and then backscattered. After demodulating and decoding, the receiver feedbacks a positive or negative feedback according to the decoding results. The tag either transmits more coded packets or a new frame based on the feedback.

3.1 Index Matrix

On the basis of Sect. 2.2, we can get one check matrix and its corresponding generator matrix through creating an index matrix. Due to the characteristics of rateless code, the way which enables us to compose infinite index matrices must be found. One considerate solution is changing the initial exponent of index matrix variables.

$$\begin{pmatrix}
 a^x b^y & a^{x+1} b^y & \dots & a^{x+m} b^y \\
 a^x b^{y+1} & a^{x+1} b^{y+1} & \dots & a^{x+m} b^{y+1} \\
 \vdots & \vdots & & \vdots \\
 a^x b^{y+n} & a^{x+1} b^{y+n} & \dots & a^{x+m} b^{y+n}
 \end{pmatrix} \tag{3}$$

The initial exponential values x and y can be substituted with any positive integer or zero. As depicted in Fig. 2, the index matrices constructed by different initial exponents have the very close BER. Therefore we can create inexhaustible and different index matrix in this way.

3.2 Generator Matrix

We assume the check matrix $\mathbf{H} = [\mathbf{H}_1, \mathbf{H}_2]$. The matrix \mathbf{H}_1 is designed based on the index (3), while the identity matrix \mathbf{I} in the middle of \mathbf{H}_2 can move up and

down freely, which can ensure that the matrix is a no-four-rings and irregular matrix. This method can not only make the coding effect close to the random LDPC, but also facilitate the calculation of its corresponding generator matrix.

After constructing the check matrix, we use Gaussian elimination line by line to rewrite the check matrix into $[\mathbf{I}|\mathbf{P}]$ form. According to Sect. 2.2, $[\mathbf{P}^T|\mathbf{I}]$ is the generator matrix, the operator T being the transpose of a matrix.

$$\mathbf{H}_1 = \begin{pmatrix} \mathbf{I}_{a^x b^y} & \mathbf{I}_{a^{x+1} b^y} & \dots & \mathbf{I}_{a^{x+m} b^y} \\ \mathbf{I}_{a^x b^{y+1}} & \mathbf{I}_{a^{x+1} b^{y+1}} & \dots & \mathbf{I}_{a^{x+m} b^{y+1}} \\ \vdots & \vdots & & \vdots \\ \mathbf{I}_{a^x b^{y+n}} & \mathbf{I}_{a^{x+1} b^{y+n}} & \dots & \mathbf{I}_{a^{x+m} b^{y+n}} \end{pmatrix} \quad (4)$$

$$\mathbf{H}_2 = \begin{pmatrix} \mathbf{I} & \mathbf{I} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{I} & \dots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ \mathbf{I} & \dots & \mathbf{0} & \mathbf{0} & & \\ \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & & \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{I} & \mathbf{I} \\ \mathbf{I} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{I} \end{pmatrix} \quad (5)$$

3.3 Rateless LDPC Encoder

First, the message sequence is multiplied by the generator matrix to form the LDPC coded sequence. We then adopt the modulation mode of on-off keying (OOK). When the WiFi signal arrives, we send the modulated symbols through the tag.

3.4 Rateless LDPC Decoder

The main processes of decoding are shown as two parts below.

LLR Calculation Firstly, the i -th modulated symbol at the tag is denoted by α_i , and the corresponding received signal is β_i . The channel LLR is then calculated as (6), where P_i^0 indicates the probability that the channel input is 0, while P_i^1 indicates the probability that the input is 1. The subscript α_i denotes each modulated symbol transmitted from the tag, i.e., the channel LLR is calculated in symbol-wise.

$$L(\alpha_i) = \ln \frac{P_i^0}{P_i^1} \quad (6)$$

with

$$P_i^0 = P(\alpha_i = 0|\beta_i) = \frac{P(\beta_i|\alpha_i = 0)}{P(\beta_i|x_i = 0) + P(\beta_i|\alpha_i = 1)}, \quad (7)$$

Algorithm 1 Rateless Decoding Algorithm

Input: $L(c_i)$
Output: The verdict of each bit \hat{x}

- 1: count=1
- 2: **while** $ACK \neq 1$ **do**
- 3: **if** $count = 1$ **then**
- 4: $(ACK, L(q_i)) = BPfunction(L(c_i))$
- 5: **else**
- 6: $L(c_i) = [L(c_i)(1 : M), L(q_i)(M + 1 : N)]$
- 7: Normalization function($L(c_i)$)
- 8: $(ACK, L(q_i)) = BPfunction(L(c_i))$
- 9: **end if**
- 10: **end while**
- 11: **if** $L(q_i) < 0$ **then**
- 12: $\hat{x}=1$
- 13: **else**
- 14: $\hat{x}=0$
- 15: **end if**

$$P_i^1 = P(\alpha_i = 1|\beta_i) = \frac{P(\beta_i|\alpha_i = 1)}{P(\beta_i|\alpha_i = 0) + P(\beta_i|\alpha_i = 1)}. \quad (8)$$

We assume that the channel is affected by AWGN with variance σ^2 . Finally, we obtain the channel LLR (9), which is fed into the iterative decoder as the priori LLR for decoding, as

$$L(\alpha_i) = \ln \frac{P(\beta|\alpha_i = 0)}{P(\beta_i|\alpha_i = 1)} = \ln \frac{e^{-\frac{(\beta_i-0)^2}{2\sigma^2}}}{e^{-\frac{(\beta_i-1)^2}{2\sigma^2}}} = \frac{1 - 2\beta}{2\sigma^2} \quad (9)$$

Rateless LDPC Decoding Algorithm. In the decoding algorithm, we use an improved algorithm based on the traditional BP algorithm which is mentioned in Sect. 2.2, hereinafter called BP function. The result of each BP function will be retained and replaced with the initial value of the next BP function. In this way, through the cooperation of infinite generator matrix described in the Sect. 2.1, we can superimpose the decision of information bits after each BP function to reduce the BER. The decoding algorithm proposed in this paper is summarized in Algorithm 1.

In Algorithm 1, BP function has two return values: a feedback signal and $L(q_i)$. Feedback signals are divided into *ACK* and *NACK*. The acknowledgment *ACK* is a signal which sends to the tag, indicates successful decoding, so that the tag transmits the following frames. On the contrary, if a *NACK* signal is sent by the receiver, the tag will continue to create a new generator matrix to encode current frame and send it to the receiver again. $L(q_i)$ is the result of BP function, which is composed of a posterior probability of information bits and a posterior probability of check bits. If decoding fails, we will retain the result of the last $L(q_i)$ information bit. In the next BP function, we will replace the

value of the information bit in the new $L(c_i)$ with retention value, and retain the value of the check bit in the new $L(c_i)$. So except that the complete LDPC code needs to be sent for the first time, only the check bit in the code needs to be sent in the future. This replacement retains the result of the last decoding, so that the value of the decision after each decoding is closer to the true value, ultimately to deliver as reliably as possible.

It should be noted that the normalization function in the above algorithm is only be used in some special cases, which will be explained in detail later.

4 Performance Evaluation

To evaluate the system performance with respect to the BER using simulations, we make the following assumptions. The backscatter link is corrupted by the AWGN with different variances. The tag totally transmits 1,000 frames with a frame length being 1,310 bits.

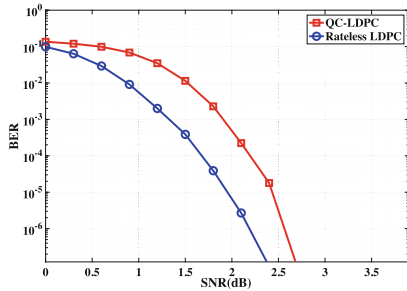


Fig. 3. The BER performance comparison between the rateless LDPC and QC-LDPC.

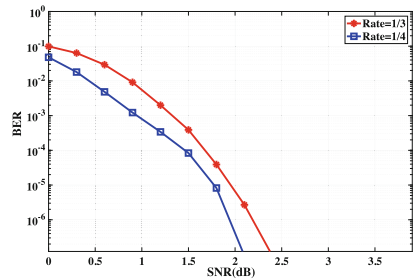


Fig. 4. The BER performance of using different coding rates for generating rate LDPC codes.

Firstly, we compare the performance of ordinary QC-LDPC and rateless LPDC. QC-LDPC adopts the check matrix form which is identical to that of the rateless LDPC codes. We choose $a = 3$, $b = 7$ as the variable values in the index matrix. For the ease of experimentation, we limited the number of retransmissions to be 1 if not mentioned. As the result, the code rate of rateless LPDC is $1/3$. The obtained simulation results are shown in Fig. 3.

As we can see, rateless LDPC outperforms QC-LDPC in all the ranges of the signal-noise ratio (SNR) in terms of BER. However, it is noted that such performance is achieved by only one retransmission. Next, the performance is evaluated by increasing the number of retransmission to show the impact of the number of retransmissions on the BER performance. We compare one retransmission with two retransmissions, which means the code rates of rateless LDPC are $1/3$ and $1/4$, respectively. Figure 4 illustrates the obtained results.

In Fig. 4, we can easily draw the conclusion that the larger numbers of retransmission, the smaller BER. Therefore, if the number of retransmission is not limited, the bit error rate can reach the optimal value, which is the best effect that rateless LDPC can achieve. Meanwhile, the code length can also affect the performance of rateless LDPC. We compared rateless LDPC with code length of 1310 and 25,390, the code rate was set at 1/3 (Fig. 5).

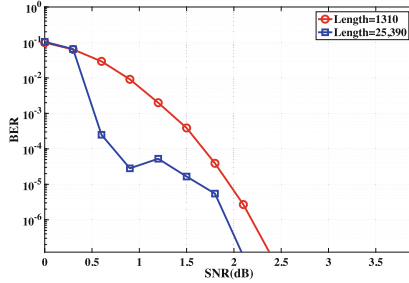


Fig. 5. Different code lengths of rateless LDPC

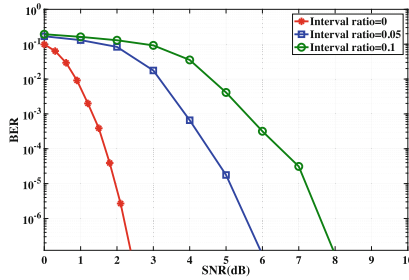


Fig. 6. Different interval ratios of WiFi packets.

In WiFi backscatter communication, according to the description in Sect. 2.1, we need to consider not only the noise interference, but also the intermittent nature of WiFi signals. Therefore, we add the simulation of WiFi intermittence. The tag sends frames as usual, if there is an interruption at some point which leads to the missing of information bits, we assume that the received bit is 0, and then decode it normally.

We compare the coding effect of rateless LDPC when the interval rate is 0, 0.05 and 0.1 respectively. Rateless LDPC adopts code length of 1310 and retransmission once, that is, the code rate is 1/3. The results are shown in Fig. 6.

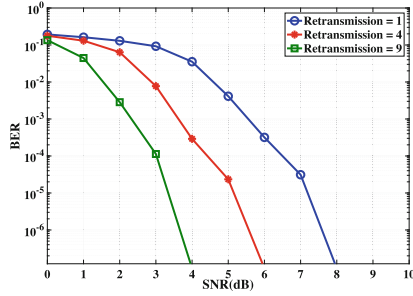


Fig. 7. Different retransmissions times with 0.1 interval ratio.

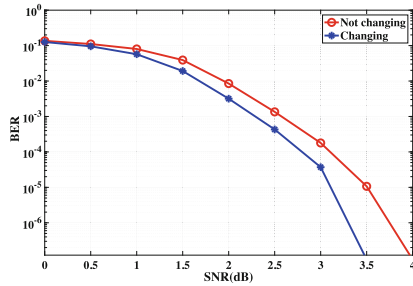


Fig. 8. Whether changing the index matrix.

It is worth mentioning that normalization method is used in this situation which mentioned in Algorithm 1. When the SNR is relatively large, it will affect the fluctuation range of the posterior probability in the BP algorithm. If it exceeds a certain range, an error value will be generated and decoding fails. Therefore, we need to use the normalization algorithm to keep the posterior probability after each decoding within a certain range, which can ensure the success of decoding. In this experiment, we adopt linear function conversion normalization to limit the posterior probability value to the range of -5 to 5 .

Next, we increase the number of retransmissions to verify whether the BER can be reduced in the case of intermittence. We selected the interval rate as 0.1 and the number of retransmissions as 1, 4 and 9 respectively. The experimental results are shown as Fig. 7.

At the same time, in order to prove the superiority of rateless LDPC coding by replacing the index matrix, we compared rateless LDPC with rateless LDPC without changing the index matrix, the later was equivalent to an improved automatic repeat request (ARQ) protocol which using the decoding algorithm mentioned in this paper. Normalization method was used in both situation and the code rate was $1/4$. The results are shown in Fig. 8. It can be concluded from that the rateless LDPC algorithm in this paper is better than the improved ARQ protocol.

5 Discussion

Among the entire experiments, although we limit the code length and retransmission times in order to reduce the complexity of the tag, rateless LDPC still shows a good performance in anti-interference and WiFi intermittence. In practical application, we will no longer limit the number of retransmission, but adjust the code length to the most appropriate value. At the receiver, we could set a BER threshold which corresponds to the desired quality of service. If the obtained BER is less than or equal to this threshold, the decoding is considered to be successful. We can even set the threshold to zero for reliable transmission. It can be predicted that rateless LDPC can shine in WiFi backscatter communication after releasing the bondage. However, the results of this work are mainly obtained by computer simulations. As a future study, we can implement our coding scheme in a prototype to evaluate the system performance in real scenarios.

6 Conclusion

In this paper, the combination of rateless coding and LDPC coding was applied to WiFi backscatter communication, which can not only effectively prevent interference and packet intermittence, but also maximize the utilization of resources. In the case of good channel conditions, it reduced the number of transmission, while in poor channel conditions, it increased the number of retransmission to achieve reliable transmission. In addition, rateless technique makes LDPC coding reach Shannon limit as much as possible, so as to ensure the excellent performance of LDPC coding.

References

1. Liu, V., Parks, A., Talla, V., Gollakota, S., Wetherall, D., Smith, J.R.: Ambient backscatter: wireless communication out of thin air. In: Proceedings of ACM SIGCOMM. ACM, New York (2013)
2. IEEE standard for information technology-telecommunications and information exchange between systems - local and metropolitan area networks-specific requirements - part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications - redline. IEEE Std 802.11-2020 (Revision of IEEE Std 802.11-2016) - Redline, pp. 1–7524 (2021)
3. Huang, J., Xing, G., Zhou, G., Zhou, R.: Beyond co-existence: exploiting WiFi white space for ZigBee performance assurance. In: The 18th IEEE International Conference on Network Protocols, pp. 305–314 (2010)
4. He, X., Jiang, W., Cheng, M., Zhou, X., Yang, P., Kurkoski, B.: GuardRider: reliable WiFi backscatter using reed-Solomon codes with QoS guarantee. In: 2020 IEEE/ACM 28th IWQoS, pp. 1–10 (2020)
5. He, C., Luan, H., Li, X., Ma, C., Han, L., Jane Wang, Z.: A simple, high-performance space-time code for MIMO backscatter communications. *IEEE Internet Things J.* **7**(4), 3586–3591 (2020)

6. Yunkai, H., Wang, P., Lin, Z., Ding, M., Liang, Y.-C.: Performance analysis of ambient backscatter systems with LDPC-coded source signals. *IEEE Trans. Veh. Technol.* **70**(8), 7870–7884 (2021)
7. Tao, Q., Zhong, C., Lin, H., Zhang, Z.: Symbol detection of ambient backscatter systems with Manchester coding. *IEEE Trans. Wireless Commun.* **17**(6), 4028–4038 (2018)
8. Zhang, P., Josephson, C., Bharadia, D., Katti, S.: FreeRider: backscatter communication using commodity radios. In: *CoNEXT*, pp. 389–401 (2017)
9. Onay, M.Y., Dulek, B.: Performance analysis of TV, FM and WiFi signals in backscatter communication networks. In: *2019 27th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4 (2019)
10. Rosenthal, J., Reynolds, M.S.: A 158 pJ/bit 1.0 mbps bluetooth low energy (BLE) compatible backscatter communication system for wireless sensing. In: *2019 IEEE Topical Conference on Wireless Sensors and Sensor Networks (WiSNet)*, pp. 1–3 (2019)
11. Daskalakis, S.-N., Kimionis, J., Collado, A., Tentzeris, M.M., Georgiadis, A.: Ambient FM backscattering for smart agricultural monitoring. In: *2017 IEEE MTT-S International Microwave Symposium (IMS)*, pp. 1339–1341 (2017)
12. Gallager, R.G.: *Low-Density Parity-Check Codes*. MIT Press, September 1963
13. MacKay, D.J.C.: Good error-correcting codes based on very sparse matrices. *IEEE Trans. Inf. Theory* **45**(2), 399–431 (1999)
14. Mackay, D., Neal, R.M.: Near Shannon limit performance of low density parity check codes. *Electron. Lett.* **32**(18), 457–458 (2013)
15. Myung, S., Yang, K., Kim, J.: Quasi-cyclic LDPC codes for fast encoding. *IEEE Trans. Inf. Theory* **51**(8), 2894–2901 (2005)
16. Zhang, H., Zhang, Z., Dai, H.: Rateless-coding-assisted multi-packet spreading over mobile networks. In: *2013 IEEE Global Communications Conference (GLOBECOM)*, pp. 5000–5005 (2013)



Design of Physical Layer Coding for Intermittent-Resistant Backscatter Communications Using Polar Codes

Xing Guo^{1,3(✉)}, Binbin Liang¹, and Xin He^{2,3}

¹ School of Computer Science and Technology, Anhui University, Hefei 230601, China
{guox,E20301219}@ahu.edu.cn

² School of Computer and Information, Anhui Normal University,
Wuhu 241002, China
xin.he@ahnu.edu.cn

³ Deqing Alpha Innovation Institute, Deqing, China

Abstract. Backscatter communications enable the connection of the large scale of the Internet of things (IoT) devices, due to their extremely low power consumption characteristic. As the number of IoT devices is increasing, the effective and reliable communication between devices becomes a key factor to offer services with the desired quality by the IoT. However, due to the impact of noise and the low power of the backscatter signal itself, the system performance is usually unreliable. To this end, in this paper, we propose an integrated cyclic redundancy check code and Polar codes (CRC-Polar) to improve the performance of the ambient backscatter communications. The performance is verified indicating by the bit error rate from the following aspects: excitation source time intervals, excitation source signal-to-noise ratios, coding rates and code lengths. We conduct extensive computer simulations using Matlab platform to verify that the designed method achieves a better enhancement of the excitation source signal transmission process. The experimental results show that our proposed CRC-Polar scheme can effectively improve the communication reliability of backscatter communication with medium and long distances and effectively reduce the influence of environmental factors on the communication quality.

Keywords: Polar code · Backscatter communication · Low power consumption · SCL algorithm

1 Introduction

With the rapid development of the Internet of things (IoT), more and more IoT devices are entering daily life, making an indispensable contribution to social

In part supported by the University Synergy Innovation Program of Anhui Province under Grant No. GXXT-2019-024.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022
L. Wang et al. (Eds.): WASA 2022, LNCS 13473, pp. 72–83, 2022.
https://doi.org/10.1007/978-3-031-19211-1_6

development while facilitating ours daily lives. Along with the rapid development of 5G communication technology, the number of IoT devices grows in an exponential manner in near future [1–8]. Facing the network formed by such a large number of IoT devices, how to achieve efficient and reliable communication between devices is a key indicator to improve the service quality. Otherwise, a good information communication exchange network cannot be formed between devices, which significantly decrease the user experience of the IoT. To tackle the challenge, the backscatter communication becomes a possible solution for connecting these large-scale IoT devices, enabling by its extremely low power consumption, at a level of μW usually.

A typical backscatter communication system contains an excitation source, a tag and a receiver. The tag does not generate any radio frequency (RF) signal to convey information. It just borrows the ambient RF signal to transmit its data from the excitation source. The receiver receives the modulated RF signal reflected by the tag. It then decodes and processes the signal to recover the data from the tag. In this process, the RF signal emitted by the excitation source may vary during one backscattering phase due to the different deploying distances of the tag, random noise and the characteristics of the excitation RF signals. As a result, the reliability of the backscatter communication usually is not acceptable. To this end, it is necessary to perform a channel encoding at the tag node to improve the reliability of the backscatter system. Inspired by the Shannon capacity achieving performance of Polar codes, we design a coding scheme to improve the reliability of long-distance operation of backscatter communications using Polar code. The system performance of the proposed backscatter communication system is evaluated from various perspectives.

The main contributions of this work are as following.

- To study the accuracy of decoding information at the receiver after the signal is transmitted from the excitation source and reflected by the tag to the receiver under four different conditions: different excitation source time interval, different excitation source signal-to-noise ratio, different excitation source code rate and different excitation source code length. The bit error rate (BER) is utilized as a performance indicator of the system performance. The lower the BER, the better the quality of service.
- We evaluate the system performances with respect to the distances between the excitation source signal and the tag, the tag and the receiver, to verify the effect by the distances.
- Simulation results show that the coding design achieves a notably good performance for a long-range operation with a low signal-to-noise ratio (SNR), which verifies the effectiveness of the coding design.

The rest of this paper is organized as follows. Section 2 presents the theoretical knowledge of polarization, convolutional and fountain code coding. Section 3 systematically presents the design of the interruption-resistant physical layer coding in this paper. The performance evaluation of the coding is given in Sect. 4. Finally, Sect. 5 concludes the whole paper.

2 Preliminary Knowledge

In this section, we introduce some vital knowledge of the system design.

2.1 Polar Codes

In 2008, Erdal Arikan first introduced the concept of channel polarization and invented a new coding method based on the channel polarization, named Polar code [9–14]. The Polar code is designed without considering the minimum distance characteristic, but using the process of channel combination and channel splitting to select the specific coding scheme, and also using probabilistic algorithms in decoding. For a polarization code of length $N = 2^n$ (n is any positive integer). It performs channel union and channel splitting using N independent copies of the channel W to obtain a new post-split channel $\{W_N^{(1)}, W_N^{(2)}, \dots, W_N^{(N)}\}$. As the code length N increases, the split channel evolves to two extremes; one part of the split channel converges to a perfect channel, i.e., a noise-free channel with channel capacity converging to 1, while the other part of the split channel converges to a completely noisy channel, i.e., a channel with a channel capacity converging to 0. Assuming that the binary input symmetric capacity of the original channel W is denoted as $I(W)$, then when the code length N tends to infinity, the proportion of split channels whose channel capacity tends to 1 is about $K = N \times I(W)$, and the proportion of channels whose capacity tends to 0 is about $N \times (1 - I(W))$. For a reliable channel with a channel capacity of 1, the message bits can be placed directly without any coding, i.e., equivalent to a coding rate of $R = 1$, while for an unreliable channel with a channel capacity of 0, frozen bits can be placed that are known in advance at both the sender and the receiver, i.e., equivalent to a coding rate of $R = 0$. The reachable coding rate $R = N \times I(W)/N = I(W)$ of the polarization code when the code length N , i.e., in theory, the polarization code can be proved to reach the channel capacity.

2.2 Convolutional Codes

Shannon proved that reliable communication can be achieved by coding when the coding rate is lower than the channel capacity. Convolutional code, as a common channel coding technique, has excellent performance. It is a type of forward error correction (FEC), which can be calculated by the convolutional formula to obtain the corresponding coding element. Each coding code element is not simply related to the current coding information, but also to the previous bit information, which is a kind of coding with memory. The constrained length and the generator polynomials determines how many bits are contributes to current coding outputs.

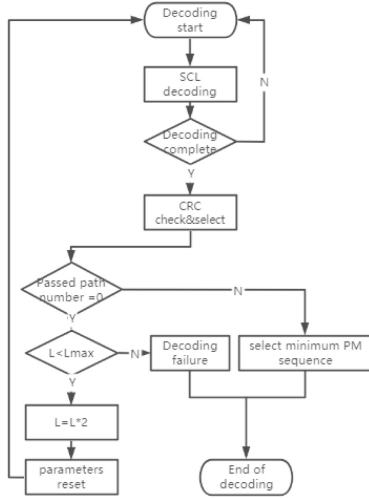


Fig. 1. The decoding process of SCL algorithm.

2.3 Fountain Code

Similar to the binary rateless code, in order to generate the simulated rateless symbols, an integer called degree is first obtained based on a predefined probability distribution function, called the degree distribution. In the next step, d different bits of information are chosen randomly and uniformly. Subsequently, the selected information bits are linearly combined in the real domain with the real weighting coefficients selected from a predefined probability distribution function called the weight distribution to generate the coded symbols.

2.4 SCL Algorithm

The successive cancellation (SC) decoding algorithm [13,14] is the successive deletion decoding algorithm, and its basic idea is to decode by judging the probability value of the likelihood of the information bits. While the successive cancellation list (SCL) decoding algorithm is an improved decoding algorithm based on the SC algorithm, which can achieve a better balance between computational and spatial complexity, and thus achieve better decoding results. The decoding process of the SCL algorithm is depicted in Fig. 1.

3 Coding Design

In this section, we describe the coding design for the backscatter system and how it works.

The system framework is illustrated in Fig. 2, which mainly contains the transmitter and receiver. Firstly, a cyclic redundancy check (CRC) code is calculated for the source information and the parity bits are appended. The CRC

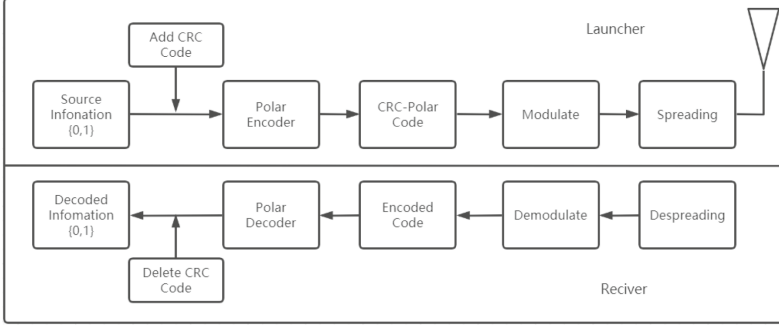


Fig. 2. Transceiver workflow of the proposed backscatter system.

coded sequence is then polarized-encoded using the Polar codes. The coded sequence is finally modulated for transmission using on-off keying (OOK) modulation by switching an RF switch. After receiving the RF signal, the receiver performs a series of computation, including demodulation, polarizes-decoding, CRC code checking. Finally, the message is reconstructed from the decoded information sequence by hard decision.

The transmitter polarizes-encodes the original message and subsequently adds CRC codes as a way to check for possible errors during transmission. The log-likelihood ratio of the polarization-encoded message sequence is then calculated, and the consequence is modulated for transmission. The decoding process at the receiver side is precisely the opposite of the encoding process.

3.1 Transmitter

The transmitter uses the channel polarization phenomenon to encode the original message sequence using the CRC-SCL algorithm, which reduces errors by the encoding. The coded information sequence is then converted to a transmitted symbol sequence utilizing the OOK modulation.

A binary input discrete memoryless channel (B-DMC) is denoted as $W : X \rightarrow Y$, where X is the set of input symbols, and Y is the set of output symbols. The transfer probability is denoted as $W(y | x)$, $x \in X$, $y \in Y$. Since the channel is a binary input, the sets $X = \{0, 1\}$, Y , $W(y | x)$ are arbitrary values. The channel after N uses for channel W can be denoted as W^N , then the channel W^N corresponds to the transfer probability $W^N(y_1^N | x_1^N) = \prod_{i=1}^N W(y_i | x_i)$. For a B-DMC W , there are two important channel parameters: the symmetric capacity as

$$I(W) \triangleq \sum_{y \in Y} \sum_{x \in X} \frac{1}{2} W(y | x) \log \frac{W(y | x)}{\frac{1}{2} W(y | 0) + \frac{1}{2} W(y | 1)} \quad (1)$$

and the Bhattacharyya parameter as

$$Z(W) \triangleq \sum \sqrt{W(y | 0)W(y | 1)} \quad (2)$$

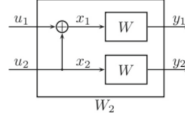


Fig. 3. Level 1 union process of the channel combining.

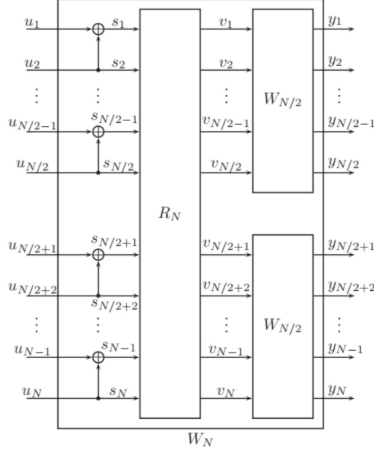


Fig. 4. The process of channel union at level N .

where $I(W)$ is a measure of the channel rate and $Z(W)$ is a measure of the channel reliability. $I(W)$ is the maximum rate at which the channel W can be reliably transmitted with equal probability input; $Z(W)$ is the upper limit of the maximum likelihood judgment error probability at which the channel W transmits only 0 or 1.

Channel polarization includes two processes: channel combining and channel splitting.

In the stage of channel combining, N independent copies of the B-DMC W are united to produce a vector channel $W_N : X^N \rightarrow Y^N$ by recursion, where N is a power of 2 to $N = 2^n, n \geq 0$. The recursion starts at level 0 ($n = 0$) with just one copy of W and defines $W_1 \triangleq W$. Level 1 unites 2 independent copies, as shown in Fig. 3.

The vector channel $W_2 : X^2 \rightarrow Y^2$ is obtained, and its transfer probability can be calculated by the following equation.

$$W_2(y_1, y_2 | u_1, u_2) = W(y_1 | u_1 \oplus u_2) W(y_2 | u_2) \quad (3)$$

The general process of channel union is shown in Fig. 4.

The two independent copies of $W_{N/2}$ jointly produce the channel W_N . The input vector u_1^N enters the channel W_N and is first transformed to $s_1^N : s_{2i-1} = u_{2i-1} \oplus u_{2i}, s_{2i} = u_{2i}, 1 \leq i \leq N/2$. R_N represents the bit-reversal sorting operation with input s_1^N and output $v_1^N = (s_1, s_3, \dots, s_{N-1}, s_2, s_4, \dots, s_N)$. And

v_1^N becomes the input of 2 independent copies of $W_{N/2}$. The mapping $u_1^N \rightarrow v_1^N$ is a linear transformation on the binary domain $GF(2)$. $u_1^N \rightarrow x_1^N$ is the input mapping from the input of the composite channel W_N to the original channel W^N , and its mapping process is also a linear transformation. Thus there is $x_1^N = u_1^N G_N$, and G_N is said to be an N -dimensional generating matrix. The transfer probabilities of channels W_N and W^N conform to the following relation

$$W_N(y_1^N | u_1^N) = W^N(y_1^N | u_1^N G_N) \quad (4)$$

where $y_1^N \in Y^N$, $u_1^N \in X^N$.

Channel splitting is the second stage of channel polarization. The composite channel W_N formed by channel union is split into N coordinate channels $W_N^{(i)} : X \rightarrow Y^N \times X^{i-1}$, $1 \leq i \leq N$ with binary inputs, and the corresponding transfer probabilities are defined as follows

$$W_N^{(i)}(y_1^N, u_1^{i-1} | u_i) \triangleq \sum_{u_{+1}^N \in X^{N-i}} \frac{1}{2^{N-1}} W_N(y_1^N | u_1^N), \quad (5)$$

where (y_1^N, u_1^{i-1}) denotes the output of $W_N^{(i)}$ and u_i denotes the input of $W_N^{(i)}$.

The transfer probabilities of odd-order splitting sub-channels and even-order splitting sub-channels can be obtained by two recursive equations. For any $n \geq 0$, $N = 2^n$, $1 \leq i \leq N/2$, there are

$$\begin{aligned} & W_N^{(2i-1)}(y_1^N, u_1^{2i-2} | u_{2i-1}) \\ &= \sum_{u_{2i}} \frac{1}{2} W_{N/2}^{(i)}(y_1^{N/2}, u_{1,o}^{2i-2} \oplus u_{1,e}^{2i-2} | u_{2i-1} \oplus u_{2i}) \cdot W_{N/2}^{(i)}(y_{N/2+1}^N, u_{1,e}^{2i-2} | u_{2i}) \end{aligned} \quad (6)$$

and

$$\begin{aligned} & W_N^{(2i)}(y_1^N, u_1^{2i-1} | u_{2i}) \\ &= \frac{1}{2} W_{N/2}^{(i)}(y_1^{N/2}, u_{1,o}^{2i-2} \oplus u_{1,e}^{2i-2} | u_{2i-1} \oplus u_{2i}) \cdot W_{N/2}^{(i)}(y_{N/2+1}^N, u_{1,e}^{2i-2} | u_{2i}) \end{aligned} \quad (7)$$

3.2 Receiver

The receiver first demodulates the signal received from the reflective tag, decodes the demodulated signal using integrated CRC and SCL (CRC-SCL) algorithm, and performs CRC checksum on the decoded information sequence. If it passes the CRC check, a hard decision needs to be done to form the reconstruction of the message from the tag.

3.3 Algorithm Design

In this paper, the polarization-decoding is performed by CRC-SCL decoding algorithm, which can largely improve the decoding performance of SCL decoding algorithm without increasing the decoding complexity too much. The decoding algorithm process is carried out in the following steps.

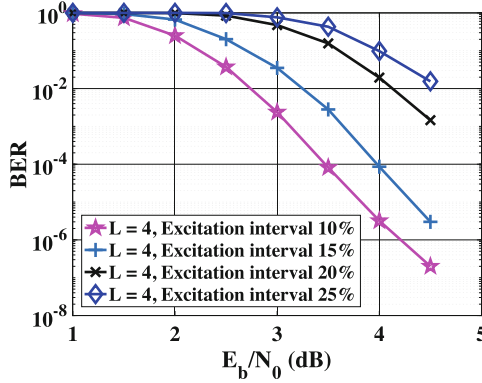


Fig. 5. BER performance with different excitation source time intervals when the reserved path $L = 4$.

1. Demodulating the received signal to form the log-likelihood ratio (LLR) which is the soft information and input into the CRC-SCL decoder.
2. It decode the received signal using the SC decoder, and if the decoded result passes the CRC check, the decoding is terminated; if the result decoded by the SC decoder does not pass the CRC check, the CRC-SCL decoder with $L = 2$ is selected to decode the signal again.
3. If there exists a CRC-SCL decoder with the list number L that can pass the CRC check, the decoding is terminated; otherwise, make $L \leftarrow 2L$ and decode the signal with CRC-SCL decoder again.
4. Repeat the above process until a result that can pass CRC checksum is produced, or L exceeds the given maximum value.

Following the above 4 steps, the decoding process is finished and the estimated message sequence is reconstructed from the decoded message sequence based on hard decision. The decoded information sequence obtained in this way achieves a good BER performance under various conditions and different parameters, which also verify the effectiveness of the coding algorithm proposed in this paper to a certain extent.

4 Performance Analysis

Matlab2018b is used as the experimental platform and simulations are performed. The SNR, code length, code rate, excitation source interval and signal attenuation are controlled to simulate the performance of this set of coding under different environments, which can be indirectly reflected by the BER.

1) Different excitation source time interval. In the case of different excitation source time interval (excitation source time interval is the ratio of the effective signal emitted by the excitation source to the overall emitting time,

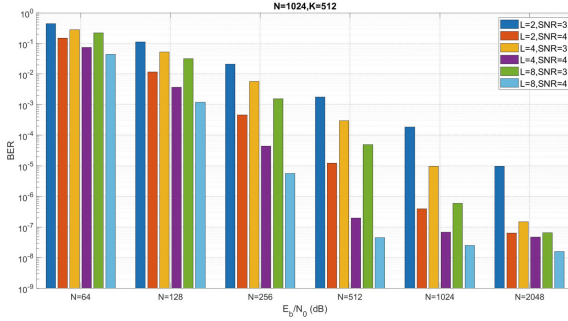


Fig. 6. BER performance of reserved paths $L = 2, 4, 8$, and SNR= 3, 4 dB with different code lengths.

the larger the excitation source time interval indicates the lower the effective information proportion; the opposite indicates the higher the effective information proportion), the excitation source time interval chosen in this paper is set as $\{10\%, 15\%, 20\%, 25\%\}$. In this range, the BER performance of the selected reserved path $L=4$ at SNR= $\{1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5\}$ is shown in Fig. 5. The BER performance decreases with increasing excitation source interval, and even at 25% excitation source interval (SNR= 4.5 dB), the proposed scheme can still maintain a good BER performance.

2) The BER performance for different SNR’s. The results of SNR= 3 dB and SNR= 4 dB with a fixed code rate $R = 0.5$ and different code lengths $N = \{64, 128, 256, 512, 1024, 2048\}$ is shown in Fig. 6 by selecting the reserved paths $L = 2, L = 4$ and $L = 8$. It can be seen that the effects of SNR and reserved path L on BER are different for each code length N , but show an overall decreasing trend.

3) Different excitation intervals with different reserved path numbers. Path numbers $L = 2, L = 4$ and $L = 8$ are selected to compare the BER performance of different excitation source time intervals under different SNR= $\{1.5, 2.5, 3.5, 4.5\}$ dB, where the obtained results as shown in Fig. 7. As the SNR continues to increase, the corresponding BER performance becomes better; and at each fixed value of SNR, the BER shows an overall dynamic decreasing trend as the number of reserved paths L continues to increase.

4) Different coding rates. The reserved paths $L = 4$ and 8 are selected to compare the BER performance at different code rates, where the code length $N = 1024$, and the results are shown in Fig. 8. At each code rate R , when the reserved path L is fixed, the performance of BER increases as the SNR keeps increasing; when the SNR is fixed, the performance of BER generally shows a decreasing trend as the number of reserved paths L increases.

5) Power loss ratio. The BER performance is then determined in different signal-to-noise ratios at fixed code length $N = 1024$, by adjusting signal power loss ratio, as shown in Fig. 9. When the code length N is fixed, the amount of signal strength attenuation increases as the distance continues to increase.

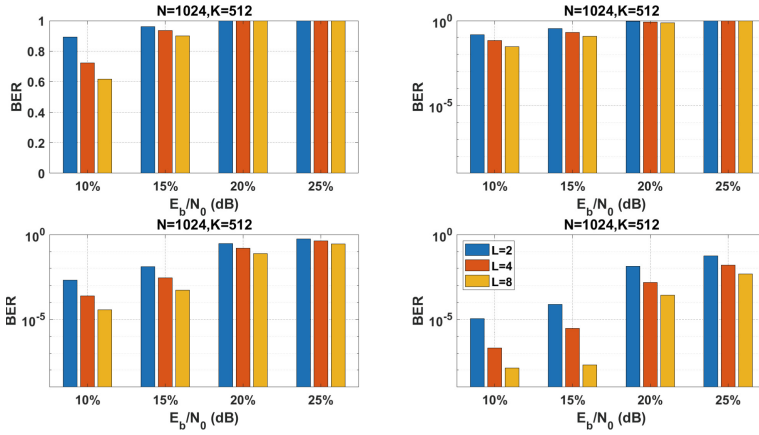


Fig. 7. BER performance at different signal-to-noise ratios for reserved paths $L = 2, 4, 8$.

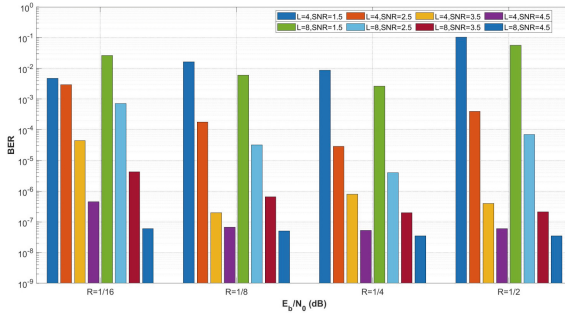


Fig. 8. BER performance of reserved paths $L = 4, 8$ at different SNR's and code rates.

At each percentage of signal attenuation, the impact on BER performance due to signal attenuation can be reduced to some extent when the SNR value is increased.

6) Joint impact of SNR and reserved paths. Finally, we evaluate the BER performance impacted by different reserved paths and different SNR with fixed code length $N = 1024$. The obtained simulation results are shown in Fig. 10. When the code length N is fixed, the increase of SNR or the number of reserved paths L is positively correlated with the performance of BER.

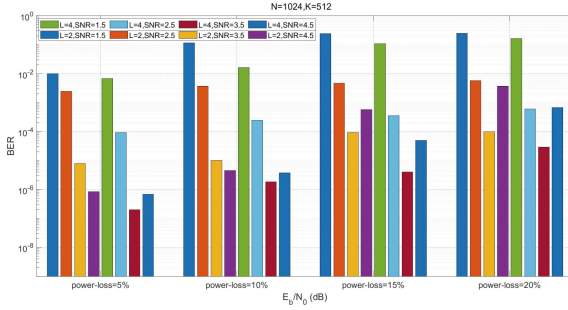


Fig. 9. BER performance under different signal fading strengths with reserved paths $L = 2, 4$.

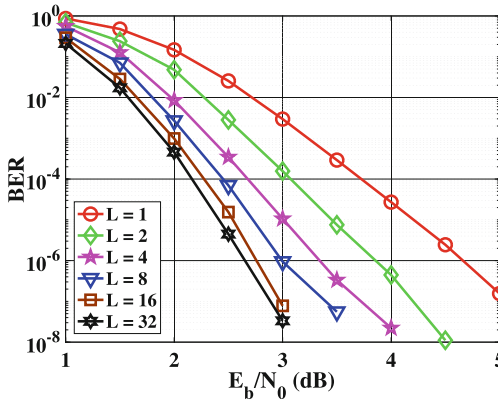


Fig. 10. BER performance with increasing SNR at different retention paths.

5 Conclusion

In this paper, the CRC-SCL algorithm was utilized to evaluate the BER of the codes generated through this coding method by comparing several different scenarios and different impact factors. After conducting numerous sets of comparison tests, it can be seen that the proposed coding method proposed can effectively solve the problem of unstable communication process caused by intermittent excitation source signal in the reflection communication process to a certain extent. The stability of the communication process in the long-distance backscatter communication process is improved, which can enhance the performance of the IoT devices when they work and thus improve the service quality.

References

1. Shannon, C.E.: A mathematical theory of communication. *Bell Syst. Tech. J.* **27**(3), 379–423 (1948)
2. Hamming, R.W.: Error detecting and error correcting codes. *Bell Syst. Tech. J.* **29**(2), 147–160 (1950)
3. Berrou, C., Glavieux, A., Thitimajshima, P.: Near Shannon limit error-correcting coding and decoding: turbo-codes. In: *Proceedings of ICC 1993 - IEEE International Conference on Communications*, vol. 2, pp. 1064–1070 (1993)
4. Gallager, R.: Low-density parity-check codes. *IRE Trans. Inf. Theory* **8**(1), 21–28 (1962)
5. Ohhashi, A., Ohtsuki, T.: Performance analysis and code design of low-density parity-check (LDPC) coded space-time transmit diversity (STTD) system. In: *IEEE Global Telecommunications Conference GLOBECOM 2004*, vol. 5, pp. 3118–3122 (2004)
6. Bharadia, D., Joshi, K.R., Kotaru, M., Katti, S.: BackFi: high throughput Wi-Fi backscatter. In: *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, SIGCOMM 2015*, pp. 283–296, New York, NY, USA. Association for Computing Machinery (2015)
7. Kellogg, B., Talla, V., Smith, J.R., Gollakot, S.: Passive Wi-Fi: bringing low power to Wi-Fi transmissions. *GetMobile: Mobile Comp. Comm.* **20**(3), 38–41 (2017)
8. Iyer, V., Talla, V., Kellogg, B., Gollakota, S., Smith, J.: Inter-technology backscatter: towards internet connectivity for implanted devices. In: *Proceedings of the 2016 ACM SIGCOMM Conference, SIGCOMM 2016*, pp. 356–369, New York, NY, USA. Association for Computing Machinery (2016)
9. Kellogg, B., Parks, A., Gollakota, S., Smith, J.R., Wetherall, D.: Wi-Fi backscatter: internet connectivity for RF-powered devices. *SIGCOMM Comput. Commun. Rev.* **44**(4), 607–618 (2014)
10. Zhang, P., Rostami, M., Hu, P., Ganesan, D.: Enabling practical backscatter communication for on-body sensors. In: *Proceedings of the 2016 ACM SIGCOMM Conference, SIGCOMM 2016*, pp. 370–383, New York, NY, USA. Association for Computing Machinery (2016)
11. Zhang, P., Bharadia, D., Joshi, K., Katti, S.: Hitchhike: practical backscatter using commodity WiFi. In: *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM, SenSys 2016*, pp. 259–271, New York, NY, USA. Association for Computing Machinery (2016)
12. Amato, F., Torun, H.M., Durgin, G.D.: RFID backscattering in long-range scenarios. *IEEE Trans. Wireless Commun.* **17**(4), 2718–2725 (2018)
13. Song, G., Yang, H., Wang, W., Jiang, T.: Reliable wide-area backscatter via channel polarization. In: *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, pp. 1300–1308 (2020)
14. Kim, T., Lee, W.: AnyScatter: Eliminating technology dependency in ambient backscatter systems. In: *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, pp. 287–296 (2020)



MEBV: Resource Optimization for Packet Classification Based on Mapping Encoding Bit Vectors

Feng Guo^{1,2}, Ning Zhang¹(✉), Qian Zou^{1,2}, Qingshan Kong¹, Zhiqiang Lv^{1,2},
and Weiqing Huang^{1,2}

¹ Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
zhangning@iie.ac.cn

² School of Cyber Security, University of Chinese Academy of Sciences,
Beijing, China

Abstract. Packet classification plays a key role in network security systems such as firewalls and QoS. The so-called packet classification is to classify packets into different categories according to a set of predefined rules. When the traditional classification algorithm is implemented based on FPGA, memory resources are wasted in storing a large number of identical rule subfields, redundant length subfields, and useless wildcards in the rules. At the same time, due to the rough processing of range matching, the rules are extended. These problems seriously waste memory resources and pose a huge challenge to FPGAs with limited hardware resources. Therefore, a field mapping encoding bit vector (MEBV) scheme is proposed, which consists of a field-splitting-recombination architecture that can accurately divide each field into four mapping preparation fields according to the matching method, field reuse rate, and wildcard ratio, and also consists of four mapping encoding algorithms to complete the length compression of the rules, to achieve the purpose of saving resources. Experimental results show that for the 1K OpenFlow 1.0 ruleset, the algorithm can achieve a significant reduction in memory resources while maintaining high throughput and support range matching, and the scheme method can save an average of 38% in memory consumption.

Keywords: Packet classification · Bit-vector · Memory compression · Mapping encoding · FPGA

1 Introduction

With the rapid development of Internet technology and the gradual increase of network security requirements, people are looking for various solutions to cope with various network attacks while the business requirements are gradually

Supported by the Chinese Academy of Sciences Project under grant NO. KGFZD-145-21-03.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022
L. Wang et al. (Eds.): WASA 2022, LNCS 13473, pp. 84–95, 2022.
https://doi.org/10.1007/978-3-031-19211-1_7

increasing. Packet classification is a technology that divides traffic into different flows based on a set of predefined rules. It can not only realize the blacklist function of separating and discarding the flow containing malicious traffic, and transparently transmitting other flows; it can also realize a similar whitelist function. As such, it is more challenging, especially in environments where packets must be processed at wire speed. And packet classification is also the core issue of OpenFlow-based software-defined networking (SDN) [1]. The ever-increasing number of fields and ever-expanding rulesets pose great challenges to practical packet classification solutions with high throughput and low memory consumption. From a hardware perspective, the main challenges for packet classification include: (1) supporting large rulesets, and (2) maintaining high performance.

In the past decade or so, software-based decision tree algorithms [2] and tuple space search algorithms [3] proposed for classical problems have been used on CPU processing platforms. However, the performance of software-based methods is limited by the CPU memory system. Hardware-based ternary content-addressable memory (TCAM) [4] solutions have also been widely used in industry due to their parallel lookup of wire-speed classification rules. But it has the disadvantages of being expensive, power-hungry, and limited capacity. Field Programmable Gate Arrays (FPGA) [5] have been widely used to process real-time networks. Algorithms based on bit vectors can make good use of the hardware parallelism of FPGAs through rule decomposition. The currently proposed bit-vector-based algorithm [6–14] can achieve high throughput by exploiting a homogeneous pipeline structure (PE) consisting of sorting processes. But as match fields and rulesets increase, the FPGA’s master clock frequency decreases rapidly and range matching is not supported. To solve the above issues, [9] divides the N-bit vector into smaller sub-vectors to improve the overall performance of classified PE in FPGA. And [12] proposed an algorithm to support range matching using precoding. However, the above approach does not reduce the required memory resources. Therefore, to achieve resource optimization while taking into account range matching and throughput, a memory-optimized scheme called MEBV based on field mapping encoding is proposed. Our contributions to this work include:

A field-splitting-recombination architecture: The architecture can accurately divide each field into four mapping preparation fields according to the matching method, field reuse rate, and wildcard ratio.

Four mapping encoding algorithm frameworks, HRME, PMME, WMME, RMME: According to the characteristics of the mapping preparation field, four different algorithms are used for mapping encoding to generate corresponding mapping vectors.

Superior memory optimization: Detailed performance evaluation of our proposed architecture on state-of-the-art FPGAs. We show in post-place-and-route results that for a 1K 12-tuple ruleset, the architecture can save 38% memory consumption on average.

Higher throughput: Compared with algorithms that support range matching, this scheme saves resources while keeping the impact on throughput within 5%, and meets the requirements for wire-speed packet classification.

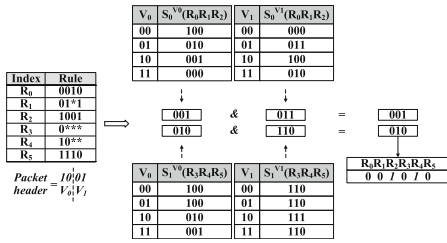


Fig. 1. Schematic diagram of stride BV algorithm. ($W = 4, s = 2$)

Field	Abbreviation	Bits
ingress port	IN_PORT	16
ethernet source address	SMAC	48
ethernet destination address	DMAC	48
Ethernet type	ETH_TYPE	16
VLAN ID	VLAN_ID	12
VLAN priority	VLAN_PRI	3
source IP	SIP	32
destination IP	DIP	32
IP protocol	PROTOCOL	8
IP ToS bits	IP_ToS	6
transmission source Port/ICMP Type	SPORT	16
transmission destination Port/ICMP Code	DPORT	16
total # of bits		253

Fig. 2. Field length and abbreviated name of OpenFlow entry.

2 Related Work

2.1 BV-Based Packet Classification

The Stride BV [7] algorithm divides a field with a bit width of W bits into $m = W/s$ subfields, where s is the length of a subfield, and the divided subrules are encoded and stored separately. This method reduces the bit width of the internal signal from W to s , which improves the throughput, but consumes more memory and is not suitable for range matching, the algorithm is shown in Fig. 1, it illustrates a Stride bv-based packet classification method. The bit vector $S_i^{V_j}$ is used to represent the matching result of V_j to the matching *subfield* _{j} corresponding to the *subruleset* _{i} . In this example, s is set to 2 and n is set to 3. For example, in Fig. 1, if the input packet header has $V_0 = 10$ in *subfield*₀ of *subruleset*₁, extract $S_1^{V_0} = 010$; this means that only rule R_4 of subruleset 1 matches the input in that subfield. The FSBV [8] algorithm is a special case, $s = 1$. When the bit width W increases, the number of pipeline stages will increase linearly with W , which will also cause a large delay in packet processing. The subsequently proposed two-dimensional pipelined stride bit vector (2D Stride BV, hereinafter referred to as 2D BV) [9], which is based on the Stride BV [7] algorithm, supports dynamic updates and further improves throughput. But the above problem still exists. Therefore, in response to the problem of memory consumption, the wildcard-removed bit-vector (WeeBV) [10] deletes the address space storing wildcards as much as possible to save memory by making full use of the characteristics of the ruleset. For the range matching problem, range bit vector encoding (RBVE) [12] has the same characteristics as Stride BV [7], and uses specially designed code to store the precomputed results in memory, which can solve the range matching problem. The subsequently proposed Range Supported Bit Vector (RSBV) [13] further improves RBVE and achieves high processing speed. However, the above methods are one of the existing problems, and cannot support range matching while saving memory.

2.2 Motivation

The process of packet classification: Given a ruleset, extracting the header field of the packet when the packet is input. Matching the header field with the corresponding fields of the ruleset, and processing the packet according to the specified action in the matched result.

The most classic is to use 5-tuple for packet classification, examining each incoming packet for the following header tuples: Source IP, Destination IP, Source Port, Destination Port, Protocol. But to accommodate today's security policies, a simple five-tuple is not enough. The OpenFlow 1.0 [15] header includes 12 fields, and the detailed description of the attributes of each field is shown in Fig. 2. By analyzing the characteristics of the OpenFlow 1.0 ruleset in [10], it is found that there are various types of search methods for different fields, and many fields have problems such as single value and a high proportion of wildcard characters: For example, the IP address field requires prefix matching, the port field requires range matching, the Ethernet type field has a fixed value and a large bit width, and the wildcard ratio of fields such as IP_ToS is high. Since the meaning of wildcards is whether the bit matches whether the bit is 0 or 1, these wildcards don't mean anything.

3 Proposed Scheme

3.1 Mapping Encoding Bit Vector (MEBV) SCHEME

MEBV is to decompose the entry into the various subfields in Fig. 3. Based on the matching form of each subfield or the number of wildcards, the F subfields are recombined into 4 mapping preparation fields. Then, different mapping algorithms are applied to each mapping preparation field for encoding to generate mapping vectors. Assuming that the lengths of the mapping preparation fields are L_1, L_2, L_3, L_4 , and the lengths of the mapping encoding fields are D_1, D_2, D_3, D_4 . Respectively, there will be $L_1 + L_2 + L_3 + L_4 \gg D_1 + D_2 + D_3 + D_4$.

For example, the OpenFlow 1.0 [15] header includes 12 fields that can be split into 12 subfields. Then analyze the characteristics of each subfield: the reuse rate of the Ethernet type and IP PROTOCOL fields is very high. Source IP and destination IP are mostly prefix matching, and the multiplexing rate is high. Source Port and Destination Port are mostly range matching. The remaining fields are exact or wildcard matches. According to the wildcard ratio of OpenFlow ruleset 12-tuple organized in MsBV [11], the fields can be divided more clearly: these fields can be arranged according to the wildcard ratio from small to large, to complete the subsequent operation of deleting wildcards. Through analysis, the detailed subfield division and mapping preparation field classification results after applying the field rearrangement technology are shown in Fig. 3.

According to the specific matching form of the field (prefix matching, range matching, exact matching, wildcard matching, etc.) or specific characteristics, the mapping preparation field adopts a specific algorithm to compress the length

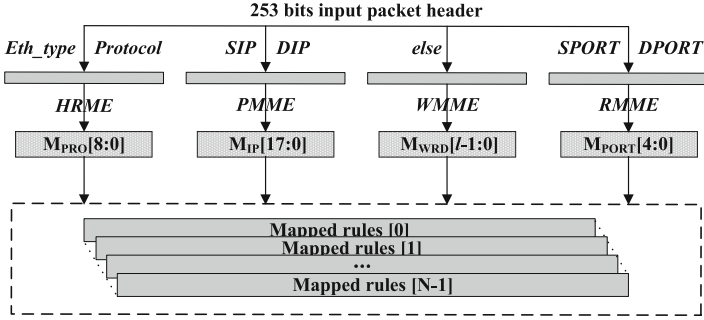


Fig. 3. MEBV scheme.

of the rule and improve the matching efficiency. The four specific algorithms of MEBV are: HRME: high reuse field mapping encoding, PMME: prefix matching field mapping encoding, WMME: wildcard matrix field mapping encoding, and RMME: range matching field mapping encoding.

Next, each mapping encoding algorithm will be introduced in detail. Please note that when this solution is applied to packet classification, rules can be customized according to the application scenario, which can achieve good scalability while compressing resources.

3.2 HRME: High Reuse Field Mapping Encoding

By analyzing a large number of rulesets and observing real traffic data, there is a serious waste of resources in the highly multiplexed field composed of the Ethernet type field and the IP protocol type field. The EtherType field has 2 bytes and 16 bits. If the Stride BV [7] algorithm is applied and divided into steps of 8, $2 * 2^8 = 512$ values are also required.

However, most of the values for this field are clustered around 0x0800 (IPv4), 0x0806 (ARP), 0x8100 (IEEE 802.1Q frame label), 0x86DD (IPv6) and wildcards, etc., while for our packet classification techniques such as OpenFlow1.0 ruleset, all belong to Ethernet II and IEEE802.3 frames. Therefore, the value of this field is more fixed. To support the blacklist function or the whitelist function, a linear mapping method can be applied to this field to save a lot of resources. The linear mapping results are shown in Table 1. The 5-bit width is selected here mainly to consider the scalability of the coding results of this field. If the application network environment is more complex, it can be expanded on this basis.

As shown above, only the five commonly used values and wildcards need to be encoded, and the other rarely used values are classified together. The method will not fail when doing a match, since this is only one of the fields. To get the correct result, the scheme needs to get the matching result of all fields.

The IP protocol type field is very similar to the Ethernet type field. The commonly used values of the IP protocol type field are 0x01 (ICMP), 0x06

Table 1. EtherType field mapping encoding result.

ETH_TYPE	V_{ETH}	M_{ETH}
IPv4	0x0800	00000
ARP	0x0806	00001
802.1Q tag	0x8100	00010
IPv6	0x86DD	00011
MPLS	0x8847	00100
else		11111

Table 2. IP protocol type field mapping encoding result.

IP_PROTOCOL	V_{PRO}	M_{PRO}
ICMP	0x01	0000
IGMP	0x02	0001
TCP	0x06	0010
UDP	0x17	0011
ESP	0x50	0100
else		1111

(TCP), 0x17 (UDP), etc., so the linear mapping method also can be used to get the encoding result, as shown in the Table 2.

3.3 PMME: Prefix Matching Field Mapping Encoding

In a set of rulesets, the consumption of redundant space can be effectively reduced by grouping DIP and SIP with the same prefix length. However, if linear classification is used, when the number of ruleset entries N increases, the space used for classification will explode, and the classification work will be extremely cumbersome. So in this scheme decided to use nonlinear classification, such as hash algorithm. By hash mapping the destination IP (or source IP) fields classified by prefix length DPL_i (or SPL_i) ($i = 0, 1, 2, \dots, 32$), a hash value of a specific length H is generated and stored in the corresponding memory. The architecture is as follows:

The ruleset has a total of N rules, which are classified according to the prefix length DPL_i ($i = 0, 1, 2, \dots, 32$). When $DPL_i < H$, the corresponding hash values are stored in the same memory, and the memory size is $n * 2^H$; when $DPL_i > H$, each P_i will maintain a piece of memory for storing the hash value generated by the IP address belonging to its own prefix length, and the memory size is also $n * 2^H$, where n represents the number of rules stored in a RAM. The processing method of SIP is the same as above. The following Fig. 4 illustrates the mapping and encoding process of the prefix matching field.

3.4 WMME: Wildcard Matrix Field Mapping Encoding

For wildcard matrix fields, the encoding of the field mapping will be different from the above. Because wildcards represent any value [11], that is, whether the field value is "0" or "1", it will be matched. So can aggregate fields with a large number of wildcards together by field rearrangement to form a wildcard matrix. When a match operation is performed, the matching result output by this matrix is 1 by default, that is, full matching. Therefore, eliminating a large number of useless wildcards in memory is also an effective means to reduce the waste of resources. A schematic diagram of applying field rearrangement techniques and rule rearrangement techniques to all rules to form a matrix of wildcards is shown in Fig. 5.

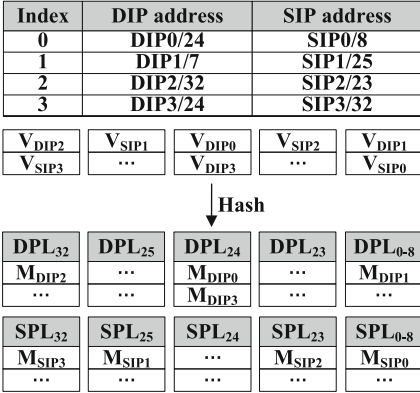


Fig. 4. Process of PMME.

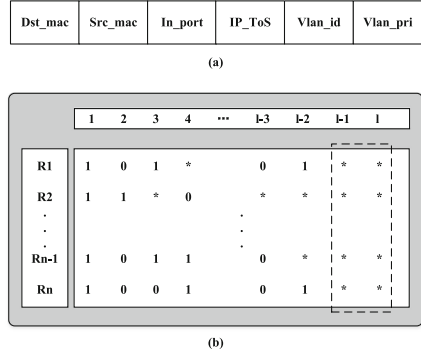


Fig. 5. Process of WMME.

3.5 RMME: Range Matching Field Mapping Encoding

The source port and destination port fields in a ruleset are usually 16-bit ranges. A 16-bit range is represented by $[LR, UR]$, where LR is the lower range limit and UR is the upper range limit. In this section, a Range Bit Vector Encoding (RBVE) [12]-like the scheme is used. In this scheme, a 16-bit range is divided into two 8-bit subranges. When performing rule matching, this classification method will cause the matching results of the latter sub-range to be related to the previous. Therefore, to complete the rule matching operation, a two-stage pipeline can be designed to place the two sub-range fields. Let V_{PORT} be a 16-bit input address and divide A into two sub-range fields in stride of 8, $V_{PORTi}, i = 0, 1$. The specific implementation process used to generate the matching signal in the RBVE [12] algorithm is shown in the following pseudocode **Algorithm 1,2** in Fig. 6. The range matching field mapping encoding process has been introduced, and the matching problem and operations will be discussed in the following chapters.

3.6 Packet Rule Matching

To better demonstrate our rule matching process, an example is as follows in Fig. 7, it shows an example of applying the MEBV algorithm to build a mapped ruleset and matching all fields of the packet to it.

We map and encode the 4 rules in the ruleset to generate mapping rules and store them in memory. When a data packet is input, the corresponding mapping encoding will be performed first to obtain the mapping result as shown in the figure. Since $s = 9$, the mapping vector produced by the eth type and protocol fields can be combined into a stride as the input address. The hash result of the source IP and the destination IP is also nine digits, which is also the length of a stride. Subsequent wildcard fields will also be divided in steps of s .

Algorithm : RMME.		
M_{PORT0} : 111 if $LR_0 < V_{PORT0} < UR_0$; 100 if $LR_0 < UR_0 \ \& \ V_{PORT0} = UR_0$; 010 if $LR_0 = V_{PORT0} = UR_0$; 001 if $LR_0 < UR_0 \ \& \ V_{PORT0} = LR_0$;	M_{PORT1} : 10 if $V_{PORT1} \leq UR_1$; 01 if $V_{PORT1} \geq LR_1$;	Match : 1 if $(M_{PORT0} = 001 \ \& \ M_{PORT1} = 01) \mid (M_{PORT0} = 100 \ \& \ M_{PORT1} = 10) \mid ((M_{PORT0} = 001 \mid 010 \mid 100) \ \& \ M_{PORT1} = 11)$

Fig. 6. Process of RMME.

When inputting a data packet, the 9-bit mapping vector of the eth type and the protocol field is used as the first set of input, all possible 9-bit mapping vectors of the destination IP are used as the second set of input, and all possible 9-bit mapping vectors of the source IP are used as the third set of input. This example is a brief introduction, so except for the fields above, the rest of the rule's fields are set to wildcards. Here, the rules are sorted by wildcard ratio, and fields with high wildcard ratios end up in a wildcard matrix. Fields with a low wildcard ratio are listed first and filled into memory according to the actual value, without mapping and encoding.

The port field is range matching, so this field adopts the RBVE algorithm. The mapping code value of this field will be directly used for matching judgment, and the composition of the mapping rule does not include this field.

3.7 Hash Collision Resolution

There will always be a problem with hash collisions when using hashing algorithms. The solution to the hash conflict in this scheme is to maintain two sets of hash algorithms. When the result of the first hash calculation produces a hash conflict, the second set of hash algorithms will be enabled. If the calculation results still conflict, the IP data will be temporarily stored, and the write address counter in this memory will be read after all the prefix length rules have been configured. Then choose the lowest address with a write address counter of 0, and force the hash result to encode this address value and store it in memory. At the same time, to minimize hash collisions under normal circumstances, the bit width of the hash value will be increased as much as possible.

4 Experimental Results and Analysis

In this section, we present the experimental setup and experimental results, which are measured in terms of space complexity, throughput, and resource consumption.

Synthetic classifiers: To test the performance of our scheme and existing techniques, we used ClassBench-ng [16], an excellent tool inherited from ClassBench [17], to generate OpenFlow1.0 rules. ClassBench-ng [16] provides torrents from real-life rules to get performance as close to practice as possible.

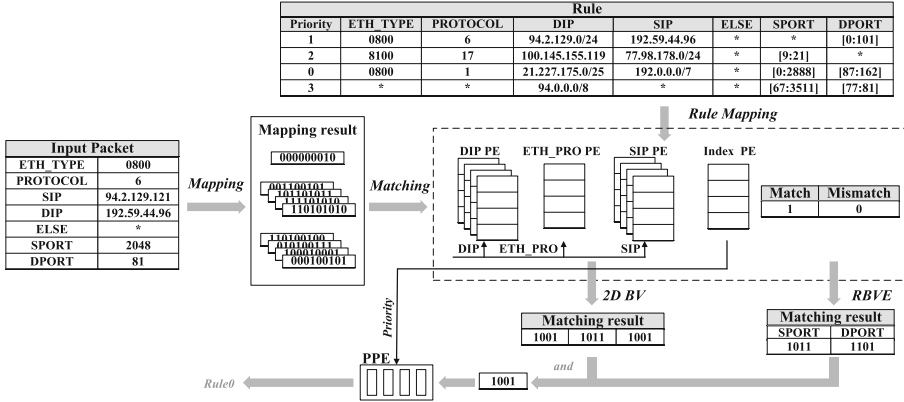


Fig. 7. MEBV: mapping encoding of rules and rules matching of packets.

The rulesets such as Accesses Control List (ACL), Firewall (FW) and IP Chain (IPC) generated by ClassBench [17] only contain traditional 5-tuple, so we use OpenFlow1.0 containing 13-tuple to prove the advanced nature of the algorithm. If excellent performance can be achieved on the OpenFlow 1.0 ruleset, so on other rulesets.

Implementation platform: The experimental environment is a Xilinx Virtex7 xc7vx690t [18] FPGA device. Limited to the experimental platform, simulation software is used to test the performance: Vivado 2018.3 and Modelsim 10.4, the results presented here are based on the post-synthesis and route performance.

4.1 Space Complexity

Assuming that there are N rules in the ruleset, the rule bit width is W , the field is divided by stride (s), and each memory stores n rules, then the space complexity of MEBV is calculated as follows. There are a total of F fields in the rule, of which f_0 fields need to complete HRME, f_1 fields need to complete PMME, f_2 fields need to complete WMME, and f_3 fields need to complete RMME. In the mapping preparation stage, f_0 and f_1 generate 8-bit mapping vector, f_2 generates l -bit mapping vector, and f_3 generates 3-bit mapping vector and 2-bit mapping vector in two stages. Therefore, the theoretical space complexity of MEBV is given by the following equation [13]

$$S_{Theory}(R) = S_{ETH_PRO}(R) + S_{IP}(R) + S_{WRD}(R) + S_{PORT}(R) \quad (1)$$

$$= (9 \times (f_0 + f_1) \times 2^9 + ((\frac{l}{s}) \times f_2 + 5 \times f_3) \times 2^8) \times N \quad (2)$$

Because the memory resources that can implement parallel operations are limited. The minimum granularity of Xilinx FPGA's [18] 18Kb and 36Kb Block RAM (BRAM) primitives is 36bit×512 and 72bit×512. Therefore, we set $s=9$,

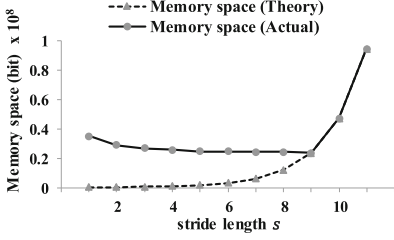


Fig. 8. Simulation of space complexity.

Algorithm	Space Complexity	Result	Range
TCAM ^[2]	$O(2(W-1)^F \times N)$	2.8×10^{35}	yes
FSBV ^[6]	$O(W \times F \times N)$	3.1×10^6	no
Stride BV ^[15]	$O((W/s) \times F \times N \times 2^2)$	9.9×10^7	no
2D BV ^[7]	$O((W/s) \times F \times N \times 2^2)$	9.9×10^7	no
RBVE ^[10]	$O((4(W/s) - 3) \times F \times N \times 2^2)$	3.9×10^8	yes
RSBV ^[11]	$O((4(W/s) - 3) \times F \times N \times 2^2)$	3.9×10^8	yes
WeeBV ^[8]	$O(((W - l_w)/s) \times F \times N \times 2^2)$	4.6×10^7	no
MEBV	$O(S_{actual}(R))$	1.6×10^7	yes

Fig. 9. Space complexity comparison.

then the memory depth required for each stage is $d=512$, and there is $F = f_0 + f_1 + f_2 + f_3$, then the formulate Summarized as

$$S_{actual}(R) = (9 \times (f_0 + f_1) + (\frac{l}{s}) \times f_2 + 5 \times f_3) \times N \times d \quad (3)$$

According to the experimental ruleset, $W = 253$, $N = 1024$, $F = 12$, $f_0 = 2$, $f_1 = 2$, $f_2 = 6$, $f_3 = 2$, $l = 25$. The value of l here is obtained from [11]. In addition, according to encoding rules, f_0 and f_2 should be 1, when calculating. In summary, the space complexity as a function of s is shown in Fig. 8. It is found that the optimal step size s in the FPGA implementation is 9, so it is decided to use a step size of 9 in 2D BV [9]. In the parallel architecture RBVE [12], since the length of the port field is 16 bits, the step size is 8.

When $s = 9$, compared with other algorithms, the space complexity is shown in Fig. 9, and it is marked whether to support range matching. Where l_w represents how many bits of wildcards are in the WeeBV [10] algorithm.

4.2 Throughput

Next, the scheme compared the throughput of multi-field packet classification. When there are $N = 1K$ rules in the ruleset, the rule bit width is $W = 253$, the $s = 9$ of 2D BV [9], the $s = 8$ of RBVE [12], and $l = 36$ rules are stored in each memory. The simulation results show that the maximum clock frequency of the MEBV algorithm is as high as 127.15 MHz. If the block RAM used in the MEBV algorithm is set as a real dual-port RAM, the throughput of the algorithm can reach 254.30 MPPS. The comparison of the MEBV algorithm with existing work is shown in Fig. 10. It can be found that the algorithm can achieve higher throughput while reducing resource consumption and effectively supporting range matching. However, the throughput of the MEBV algorithm is 14% lower than 2D BV and 1.2% lower than RBVE. The main reason is that the algorithm is executed in parallel by 2D BV and RBVE, and the throughput is determined by the smallest channel. Furthermore, due to the mapping encoding of this algorithm, this will lead to a certain decrease in the clock frequency.

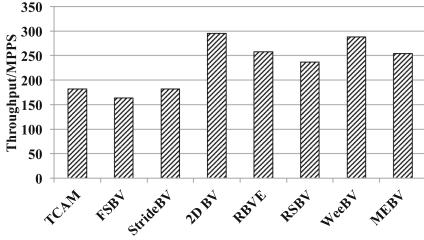


Fig. 10. Throughput comparison between algorithms.

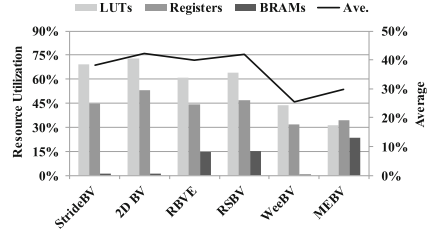


Fig. 11. Comparison of resource utilization between algorithms.

4.3 Resource Consumption

Then, we mainly focus on the FPGA resource consumption of the proposed MEBV algorithm. $N = 1024, W = 253, 2D\ BV$ [9] for $s = 9$, $RBVE$ [12] for $s = 8$, $l = 36$. The FPGA resource consumption comparison of different flow classification algorithms is shown in Fig. 11. The MEBV uses mapping encoding to reduce the length of storage rules, which can save a lot of register resources and LUT resources, but because it supports range matching and the results of mapping encoding are stored in BRAM, the consumption of BRAM resources increases. But in general, a large number of register resources are often built-in FPGA chips (Virtex 7 xc7vx690t FPGA [18] has built-in 52Mb BRAM resources). In this scheme, the BRAM resource consumption rate is 23.3%, while the LUT resource consumption rate is 43.8%, so the extra BRAM cost will not become a bottleneck, and the saved resources can support more strategies. To make the experimental results more representative and comparative, the resource utilization here is the result after placement and routing.

The broken line in the figure represents the average resource utilization, which can better reflect the superiority of our scheme. It is worth noting that the algorithm has the smallest average resource consumption among the algorithms that support range matching. That is to say, under the premise of the same resources, the scheme can support more strategies, which undoubtedly further reduces the bottleneck caused by resources.

5 Conclusion

In this paper, we propose MEBV, a memory-optimized scheme based on field mapping encoding bit-vectors, which achieves resource optimization while considering range matching and throughput. Our proposed solution can save 32.6% of LUT resources and 12.4% of register resources compared to state-of-the-art algorithms [13] that support range matching, with throughput impact remaining within 2%. Compared with the matching algorithm [9], 41.6% of LUT resources and 18.4% of register resources can be saved, and the impact of throughput remains within 12%. Meet the requirements of wire-speed packet classification.

References

1. McKeown, N., et al.: OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Comput. Commun. Rev.* **38**(2), 69–74 (2008)
2. Erdem, O., Bazlamaççi, C.F.: Array design for Trie-based IP lookup. *IEEE Commun. Lett.* **14**(8), 773–775 (2010)
3. Song, H., Turner, J., Dharmapurikar, S.: Packet classification using coarse-grained tuple spaces. In: 2006 Symposium on Architecture For Networking And Communications Systems, pp. 41–50. IEEE (2006)
4. Yu, F., Katz, R.H., Lakshman, T.: Efficient multimatch packet classification and lookup with TCAM. *IEEE Micro* **25**(1), 50–59 (2005)
5. Fu, W., Li, T., Sun, Z.: FAS: using FPGA to accelerate and secure SDN software switches. *Secur. Commun. Netw.* **2018** (2018)
6. Lakshman, T., Stiliadis, D.: High-speed policy-based packet forwarding using efficient multi-dimensional range matching. *ACM SIGCOMM Comput. Commun. Rev.* **28**(4), 203–214 (1998)
7. Ganegedara, T., Prasanna, V.K.: StrideBV: single chip 400G+ packet classification. In: 2012 IEEE 13th International Conference on High Performance Switching and Routing, pp. 1–6. IEEE (2012)
8. Jiang, W., Prasanna, V.K.: Field-split parallel architecture for high performance multi-match packet classification using FPGAS. In: Proceedings of the Twenty-First Annual Symposium on Parallelism in Algorithms and Architectures, pp. 188–196 (2009)
9. Qu, Y.R., Prasanna, V.K.: High-performance and dynamically updatable packet classification engine on FPGA. *IEEE Trans. Parallel Distrib. Syst.* **27**(1), 197–209 (2015)
10. Li, C., Li, T., Li, J., Li, D., Yang, H., Wang, B.: Memory optimization for bit-vector-based packet classification on FPGA. *Electronics* **8**(10), 1159 (2019)
11. Shi, Z., Yang, H., Li, J., Li, C., Li, T., Wang, B.: MsBV: a memory compression scheme for bit-vector-based classification lookup tables. *IEEE Access* **8**, 38 673–38 681 (2020)
12. Chang, Y.-K., Hsueh, C.-S.: Range-enhanced packet classification design on FPGA. *IEEE Trans. Emerg. Top. Comput.* **4**(2), 214–224 (2015)
13. Zheng, L., Jiang, J., Pan, W., Liu, H.: High-performance and range-supported packet classification algorithm for network security systems in SDN. In: 2020 IEEE International Conference on Communications Workshops (ICC Workshops), pp. 1–6. IEEE (2020)
14. Zhou, Q., Yu, J., Li, D.: TSSBV: a conflict-free flow rule management algorithm in SDN switches. In: 2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring), pp. 1–5. IEEE (2021)
15. Heller, B.: OpenFlow switch specification, version 1.0. 0. Wire, December 2009
16. Matoušek, J., Antichi, G., Lučanský, A., Moore, A.W., Kořenek, J.: ClassBench: recasting ClassBench after a decade of network evolution. In: 2017 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), pp. 204–216. IEEE (2017)
17. Taylor, D.E., Turner, J.S.: ClassBench: a packet classification benchmark. *IEEE/ACM Trans. Network.* **15**(3), 499–511 (2007)
18. XA Programmable: Series FPGAS overview 7



NT-RP: A High-Versatility Approach for Network Telemetry Based on FPGA Dynamic Reconfigurable Pipeline

Deyu Zhao^{1,2,3}, Guang Cheng^{1,2,3(✉)}, Yuyu Zhao^{1,2,3}, and Ruixing Zhu^{1,2,3}

¹ School of Cyber Science and Engineering, Southeast University, Nanjing, China
{dyzhao, chengguang, yyzhao, 220215334}@seu.edu.cn

² International Governance Research Base of Cyberspace, Southeast University, Nanjing, China

³ Jiangsu Provincial Engineering Research Center of Security for Ubiquitous Network, Nanjing, China

Abstract. Network telemetry provides more accurate and reliable services for intelligent network control by pushing fresh status information actively with help of data plane. However, most existing network telemetry methods are difficult to be deployed effectively in business environment due to the lack of runtime reconfigurability, huge time-space overhead, and high probability of information loss. In this work, we propose a high-versatility approach for network telemetry based on FPGA dynamic reconfigurable pipeline called NT-RP to maintain the balance between the accuracy of the measurement and the overhead in different scenarios. NT-RP can change the processing logic in runtime to obtain different network measurement spontaneously desired by users. Benefiting from distributed cyclic storage strategy and telemetry function integration mechanism, NT-RP can greatly reduce the overhead during measurement and mitigate the telemetry information missing problem caused by packet loss. The implementation of NT-RP in FPGA is evaluated in a real network testbed which consists of a few programmable nodes. Experimental results show that the influence of NT-RP in large traffic scenarios is less than 1%. It is not only able to successfully change the telemetry task during operation, but also perform more accurate network measurements with little telemetry information occupancy.

Keywords: Network telemetry · FPGA-based pipeline · Performance evaluation · Runtime reconfigurability

1 Introduction

Network telemetry is a key technology in the field of network measurement. It improves the real-time and accuracy of network measurement by using data planes to directly drive the network measurement process. However, issues such as huge overhead and the lack of telemetry information caused by packet loss seriously affect telemetry performance and impose additional challenges for telemetry data collection and processing.

In addition, most of the existing network telemetry methods are implemented by P4 [6]. Although the P4 language is well suited for data-plane related network processing, it has the disadvantage of a relatively homogeneous programming framework that does not allow for flexible reconfiguration of multiple processing logics, which results in its inability to support a network telemetry method for variable tasks. In fact, using the flexibility and scalability of the programmable data plane to form a multi-dimensional view of the network state has become one of the goals of network telemetry.

Pipeline is a temporally serial, spatially parallel technique that splits the entire digital processing logic into multiple modules. Considering that the parallel processing characteristic of FPGA is well suited for pipeline architecture, FPGA-based pipeline is becoming one of the preferred architectures for low-latency, high-throughput network processing systems. Dynamic reconfigurable pipeline is a further approach to apply the FPGA dynamic reconfiguration idea to the pipeline, which can dynamically adjust the pipeline modules to accomplish different functions [10], making the whole network processing system with good scalability while guaranteeing the wire-speed processing performance. Currently, dynamic reconfigurable pipeline is mainly used to design general customizable network processing platforms and no researcher has applied this technique for network telemetry.

In this paper, we design a network telemetry-oriented FPGA dynamically reconfigurable pipeline NT-RP, which can flexibly obtain different link-level network telemetry information with lower time-space overhead. The research contributions of this paper can be summarized as follows:

1. We design an FPGA dynamic reconfigurable pipeline named NT-RP for network telemetry. NT-RP accomplishes the telemetry tasks by considering them as the result of combining multiple fine-grained telemetry metadata (NTM) which can be directly measured. NT-RP can complete different telemetry tasks according to user requirements by adjusting the NTM calculation modules in runtime. We propose a reasonable NTM set to optimize the measurement method for link-level network characteristics based on NT-RP.
2. We implement a distributed NTM cyclic storage strategy. On the basis of assisting NT-RP to accurately complete various telemetry tasks, this strategy can greatly cut down the storage overhead of network telemetry information in packets. In addition, benefiting from the periodic data memory capability of this strategy, NT-RP can mitigate the impact of telemetry information missing due to packet loss and improve the system robustness.
3. We propose a method to better integrate telemetry functions with packet forwarding. By using the parallelism of FPGAs, NT-RP enables most of the telemetry processing logic as a part of the packet forwarding operation, eliminating the extra time overhead caused by telemetry. The integration method allows network devices loaded with NT-RP to perform telemetry with little impact on packet forwarding performance.

2 Related Work

Network telemetry provides more accurate and reliable services for intelligent network control by using the data plane to actively push network status information. In recent years, many P4-based network telemetry has been demonstrated for different purposes in [1, 2, 7]. The authors of [1] showed how to implement network telemetry with a very low amount of information added to each packet. LossSight was developed in [7], which solved the problem of high-speed real-time telemetry information loss caused by network event packet loss. More recently, a P4-based selective telemetry method has been proposed in [2] named sINT. However, sINT is not fully runtime-programmable because P4-based network telemetry methods are too fixed to change the packet processing logics at runtime. Then, Tang et al. [8] proposed Sel-Int that can dynamically adjust the type of telemetry data obtained with runtime programmability based on POF. Nevertheless, Sel-Int cannot change the calculation logic of the pipeline at runtime to obtain only the telemetry information needed for a specific task and carry it into the packet, which still generates a large amount of telemetry data as traditional telemetry methods. In addition, [3, 9, 10] provided a new idea for customizing network processing functions by exploiting the reconfigurability of the FPGA pipeline. However, none of them has application development and integration for network telemetry. [5] proposed an FPGA-based network measurement integration approach which is not applicable in multi-tasking measurement scenarios.

The above researches have already contributed in solving some typical network telemetry problems. However, deploying network telemetry on network processing platforms faces the difficulties of weak systematicity, low data collection efficiency, limited task category, and high time-space overhead. Therefore, we design NT-RP based on two optimization mechanisms to achieve runtime multi-task switching capability with low time-space overhead for network telemetry.

3 Architecture and Methodology of NT-RP

3.1 Overview

The NT-RP is built on a programmable node that contains FPGA for packet processing and CPU for configuration works. As shown in Fig. 1, NT-RP consists of five processing stages. The modules in the first three stages are fixed while some modules which can be configured through interface in the last two stages contain multiple parallel subsets to be selected.

Input Stage. The first stage is the input stage, where the incoming processor module is mainly responsible for GMII receiving and calculating the packet parameters needed for the telemetry such as packet input timestamp (IT) and total packet length.

Packet Classification Stage. After passing through the GMII receiver, the packet is sent to the packet classification stage which contains two fixed modules. If the packet is classified as a data packet, the common processor module

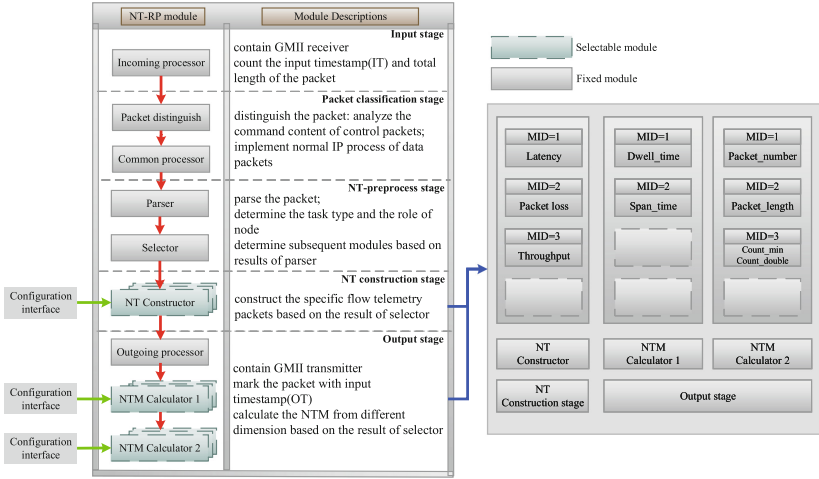


Fig. 1. NT-RP packet processing architecture

will perform the normal IP operation by decreasing the `hop_limit` by 1. On the contrary, the common processor module will parse and store the type of task represented by the instruction from clients in an instruction register.

NT-Preprocess Stage. This stage includes two modules which are mainly responsible for determining the role of the node and selecting the telemetry task. The Parser module determines whether the node is a source node based on whether the instruction register is written or not, and extracts the task type based on the contents of the instruction register or the packet (if it is not a source node). The Selector module determines the subsequent processing logic based on the parser module's result in the form of a series of module ID (MID) sent to the subsequent stages for selecting the corresponding modules. As shown in Fig. 1, there are three modules containing multiple parallel subsets after the NT-preprocess stage, and each subset is identified by a MID.

NT-Construction Stage. The NT-packet construction stage will be activated only when the node is determined as a source node. As shown in Fig. 1, we load different NT-constructors in this stage to construct telemetry packets for different types of telemetry tasks. This stage select the NT-constructor according to the result of previous modules and embed the *MeasurementID* representing the telemetry task in the specified position of the packet, so that the telemetry task type can be judged by the NT-preprocess stage when the telemetry packet enters the intermediate node.

Output Stage. This final stage in the pipeline handles the operations related to the packet forwarding and calculations of NTM. Packets are marked with an output timestamp (OT) when passing through the outgoing processor module, from which we get the three most basic packet values: IT, OT and packet length. Then, the output stage selects the appropriate NTM calculators based on the

MID to calculate the NTM we need by using the three most basic packet values. As shown in Fig. 1, we divide the NTM calculator into two modules, which can improve the overall throughput of NT-RP and provide more multidimensional telemetry services because the calculations of some specific NTM involved in some telemetry tasks require the use of NTM calculator 1's results.

3.2 Distributed NTM Cyclic Storage Strategy

We propose a distributed NTM cyclic storage strategy to alleviate the telemetry invalidity caused by network packet loss and reduce the occupancy of telemetry information in packets.

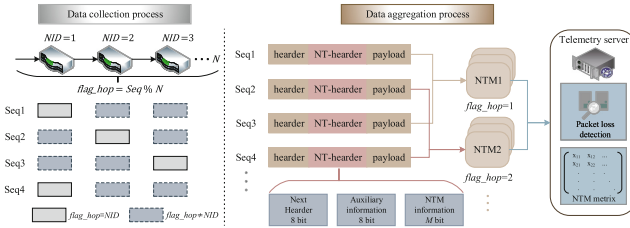


Fig. 2. Distributed NTM cyclic storage strategy

As shown in Fig. 2, the distributed NTM cyclic storage strategy is divided into two phases. In the data collection phase, each telemetry packet is assigned a $flag_hop$ to determine at which node it should collect telemetry information based on the number of measurement nodes and the sequence number of the packet. We also assign an NID to each node for identification. If $flag_hop \neq NID$, the NTM calculated based on this telemetry packet is only superimposed in the specific register Reg_n (n represents the type of NTM). Otherwise, the telemetry packet takes all previously NTM information stored in Reg_n into the NT-header before clearing Reg_n to zero. Then, the telemetry packets with the same $flag_hop$ will sum the value of Reg_n they carry to get x_{mn} to form a NTM matrix at the telemetry server which will be used to calculate the results of telemetry tasks. ($N_{collect}$ is the number of telemetry packets whose $flag_hop$ is equal to the NID of node m)

This distributed strategy can greatly reduce information overhead bought to packets for all types of telemetry tasks. Assuming that there are N nodes, as shown in Fig. 2, in addition to the M -bits NTM information occupancy, only 16 bits (auxiliary information contains $flag_hop$ and other flag bits required for tasks) need to be carried while the traditional network telemetry method [6] at least needs $M \times N$ bits to be allocated in the packet under the same measurement environment. Therefore, the telemetry information occupancy is reduced in NT-RP by at least (%):

$$J = \frac{MN - (16 + M)}{1500 \times 8} \quad (1)$$

In addition, this strategy enables NT-RP to have periodic memory of data by storing the telemetry results of each packet periodically in Reg_n . We assume that the inevitable random packet loss rate of each hop caused by node's hardware properties is θ_m , the k -th node has a network event at some point in time. If the network event does not result in event packet loss, NT-RP can store the telemetry state information associated with the event in node k as long as the packet with $flag_hop \neq NID$ is not lost until it reaches the $(k + 1)$ -th node, so the probability of each packet causing the event information lost is:

$$P_{ntrp_loss} = \begin{cases} 1 - \prod_{m=1}^k (1 - \theta_m), & flag_hop \neq NID \\ 1 - \prod_{m=1}^N (1 - \theta_m), & flag_hop = NID \end{cases} \quad (2)$$

However, It is obvious that the probability of each telemetry packet resulting in event information loss with the traditional telemetry method is always similar to the P_{ntrp_loss} in the case of $flag_hop = NID$, which confirms that NT-RP can reduce the probability of missing telemetry information problem due to the inevitable random packet loss. Also, if the network event causes packet loss at node k , for traditional INT, it is impossible to locate the exact location of the malicious event because no telemetry information can be collected during the event duration. NT-RP can alleviate this problem to some extent. Assuming that the packet loss event occurs during a telemetry process C , we can locate this event by the significant change of some values. To define "significant change", we first compare the time interval $T_{interval}$ between any two consecutive telemetry packets reaching the telemetry server with a time interval threshold $T(C)$, if $T_{interval} > T(C)$, we need further calculate the difference R_{dif} between the Reg_n which is carried in any one of the subsequent N packets and its forward packet with the same $flag_hop$. Let $R(C)$ be the threshold for C , we can determine that the node represented by the $flag_hop$ of the last packet satisfying $R_{dif} \leq R(C)$ among the N packets is the previous node of the node where the packet loss event occurs. With these two stages of detection, NT-RP can determine the location of packet loss in time during the measurement process.

3.3 Telemetry Function Integration Mechanism

As shown in Fig. 3, NT-RP takes advantage of the FPGA's parallelism and moves the NTM calculators to the end of the pipeline to be integrated as part of the GMII interface logic in the output stage, which maximally eliminates the extra time overhead that the telemetry task may bring to the normal forwarding of the packet.

There are two phases in this integration mechanism. In NTM calculation phase, the NTM calculators work in parallel with the GMII transmitter module. Specifically, the GMII transmitter module performs the bit-width conversion of the packet while the output of each clock cycle is accompanied by the logical advance of the NTM calculators. Since the NT header is preceded by a 40-byte fixed-length IP header and the time required for each NTM calculator to complete the logical calculation is only 1 to 2 clock cycles, the NT-RP is fully

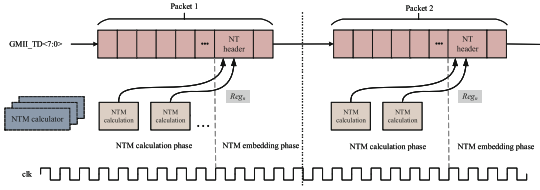


Fig. 3. Telemetry function integration mechanism

capable of completing various NTM calculations and storing them in Reg_n during the time when the fixed-length header processing is performed.

In the NTM embedding phase, the GMII transmitter module starts to send the NT-header and its subsequent packet contents to the link, while inserting the value of Reg_n calculated in the previous phase into the specified position of the NT-header.

3.4 NT-RP Based Network Telemetry Methods

To form a more reasonable and complete view of the network state, we have configured various telemetry tasks in NT-RP, including three flow-oriented telemetry tasks, namely one-way average latency, packet loss rate, flow throughput, and link-oriented link available bandwidth. We have introduced four common NTM that cover most telemetry tasks:

Packet_number: The number of telemetry packets. NTM calculator maintains the Reg_{number} to count the number of telemetry packets.

Packet_length: The length of the packets of the measured flow. NTM calculator maintains the Reg_{length} to count the amount of data (bytes) transferred by the flow.

Dwell_time: the total delay of the packet in the node. The difference between IT and OT representing the processing and queuing delay while the transmission delay can be expressed by $Packet.length/V$, where the constant V represents the hardware processing speed. Reg_{dwell} is used to store the sum of these three types of delay.

Span.time: the time interval between two consecutive telemetry packets. We obtain this value needed to be stored in Reg_{span} by calculating the difference of the OT of two consecutive telemetry packets.

Assuming that the total number of telemetry packets is N_p and the number of telemetry packets whose $flag.hop$ is equal to d is $N_{collect}$, we can use f_1 to f_3 to get the telemetry result of link d easily:

Flow one-way average latency: The average latency of a flow on a specific link. Reg_{dwell_i} represents the value of Reg_{dwell} carried in the i -th telemetry packet with the $flag.hop == d$.

$$f_1 = \frac{\sum_{i=1}^{N_{collect}} Reg_{dwell_i}}{N_p} \quad (3)$$

Flow packet loss: The packet loss for a flow on a specific link. $Reg_{number.i}$ represents the value of Reg_{number} carried in the i -th telemetry packet with the $flag_hop == d$.

$$f_2 = N_p - \sum_{i=1}^{N_{collect}} Reg_{number.i} \quad (4)$$

Flow average throughput: The average throughput of a flow on a specific link. $Reg_{length.i}$ and $Reg_{span.i}$ respectively represent the value of Reg_{length} and Reg_{span} carried in the i -th telemetry packet with the $flag_hop == d$.

$$f_3 = \frac{\sum_{i=1}^{N_{collect}} Reg_{length.i}}{\sum_{i=1}^{N_{collect}} Reg_{span.i}} \quad (5)$$

However, some telemetry tasks such as link available bandwidth cannot be directly completed from the most basic NTM above. Therefore, benefiting from the FPGA's ability to provide accurate hardware timestamps, we designed a more efficient available bandwidth measurement model based on NT-RP and updated the NTM set.

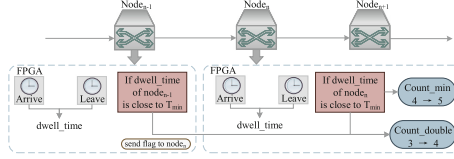


Fig. 4. Available bandwidth measurement method based on NT-RP

Inspired by the available bandwidth measurement model SMART [4], the available bandwidth measurement based on NT-RP is shown in Fig. 4. We send a certain number of short telemetry probes randomly and measure the minimum residence time T_{min} of the probes (i.e., the average $Dwell_time$ of the probes in the node without any background traffic). We consider the node to be idle if the probe's $Dwell_time$ is very close to T_{min} . Otherwise, the node is considered busy. Therefore, we maintain register $count_min$ in NT-RP to count the number of probes for whose $Dwell_time$ is close to T_{min} . In addition, to exclude the effect of certain deterministic cases on Monte Carlo randomness, we maintain the register $count_double$ to store the number of probes whose $Dwell_time$ is close to T_{min} for both the node and its preceding node. We obtain the idle rate $Free.b_i = count_double_i / count_min_{i-1}$ of link i and find the true link-level available bandwidth $Avail.bw_i = C_i \cdot Free.b_i$: (C_i is the maximum bandwidth capacity of link i , $count_min_i$ and $count_double_i$ respectively represent the values of the corresponding registers in node of link i).

This method solves the problem of SMART's poor accuracy of delay acquisition at the software level, and effectively balances the accuracy of link available bandwidth measurement with the availability of NT-RP services.

In summary, we currently have five types of NTM calculation modules loaded in NT-RP, two of which are *Dwell_time* and *Span_time* configured in NTM calculator 1 for the time dimension, and the other three are *Packet_number*, *Packet_length*, and *count_min/count_double* configured in NTM calculator 2 for the packet feature dimension. Table 1 lists the NTM used by different telemetry tasks in NT-RP, and defines the modules to be selected in the last two stages of the pipeline according to the different task types and node roles.

Table 1. The NTM involved in telemetry tasks and the corresponding processing logic (M1: MID of NT-processor, M2: MID of NT-meta calculator1, M3: MID of NT-meta calculator2, -: empty)

Telemetry type	Telemetry task	Node role	NTM needed	M1	M2	M3
Flow-telemetry	One-way average latency	Source	<i>Dwell_time</i>	1	1	–
	Packet loss	source	<i>Packet_number</i>	2	–	1
	Average throughput	Source	<i>Packet_length</i> , <i>Span_time</i>	3	2	2
	One-way average latency	Not-source	<i>Dwell_time</i>	–	1	–
	Packet loss	Not-source	<i>Packet_number</i>	–	–	1
	Average throughput	Not-source	<i>Packet_length</i> , <i>Span_time</i>	–	2	2
Link-telemetry	Available bandwidth	Source or not-source	<i>count_min</i> , <i>count_double</i>	–	1	3

4 Performance Evaluation

The NT-RP prototype is deployed on three programmable nodes equipped with an FPGA clocked at 125 Mhz and two 1000 Mbps ethernet interfaces. The pipeline module in programmable nodes uses FAST architecture as [11] which meets the performance and flexibility requirements of network device function expansion. Three programmable nodes loaded with NT-RP are configured as telemetry source node, intermediate node and tail node respectively.

We evaluate the performance of NT-RP by conducting four groups of experiments, these experiments validate the performance of NT-RP from four perspectives: accuracy of telemetry results, integration with normal forwarding, system robustness in the face of packet loss and runtime reconfigurability.

Accuracy of Telemetry Results. Figure 5(a–c) shows the measurement results for the four telemetry tasks. As expected, the flow one-way average latency which is consistent with the processing rate of the FPGA increases with the increase of packet size while the loss rate measured by NT-RP is always maintained at about 0.1% for different background traffic intensities, which is very close to the real value. In addition, the measured values of flow throughput at different packet lengths are very accurate compared to true throughput of flow.

Figure 5(d) compares the available bandwidth measured by NT-RP and SMART for different network utilization cases. The relative error of NT-RP measurement results is always maintained within a good range and less than that of SMART.

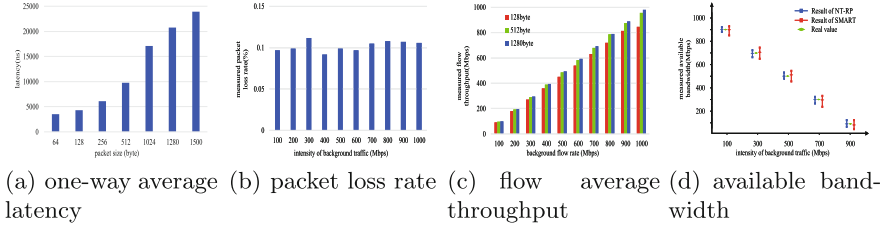


Fig. 5. Results of various telemetry tasks generated by NT-RP

Integration with Normal Forwarding. To demonstrate that the telemetry function of NT-RP is great integrated with the normal forwarding of packets, we compare the average packet forwarding latency and rate between the programmable node loaded with the NT-RP telemetry pipeline and that acting only as a normal router. During the experiment, we set the background traffic rate to 1000Mbps and change the packet size to obtain the results.

Figure 6(a) shows that nodes loaded with NT-RP suffer from slightly longer average latency of 8ns in packet forwarding because most of the additional telemetry processing logic has been integrated into the normal forwarding, and only the NT-processor module, which cannot be integrated, causes a small additional processing delay for packet forwarding. Moreover, NT-RP has only a small impact on the normal packet forwarding rate. In Fig. 6(b), with NT-RP loaded, the loss of the packet forwarding rate decreases as the packet size increases and drops below 1% when the packet size reaches 1024 bytes.

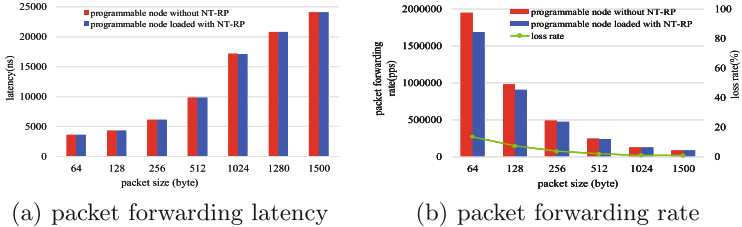


Fig. 6. Data plane performance: NT-RP loaded node vs. base node

System Robustness in the Face of Packet Loss. We deploy this packet loss location detection mechanism into the telemetry server and determine two thresholds based on extensive experiments: $T(C) = 50$ ms and $R(C) = 2\%$. We inject a large amount of traffic to the intermediate node at some time during the telemetry process causing its congestion and generating packet loss. As shown in Fig. 7(a), after receiving the packet with $T_{interval}$ over 50 ns, the telemetry server calculates the R_{dif} for the three subsequent packets including this packet. It is obvious from Fig. 7(b) that the $flag_hop$ of the last packet whose R_{dif} does

not exceed 2% of the three packets is equal to 1, so the location where the packet loss occurs is the intermediate node with *flag_hop* of 2.

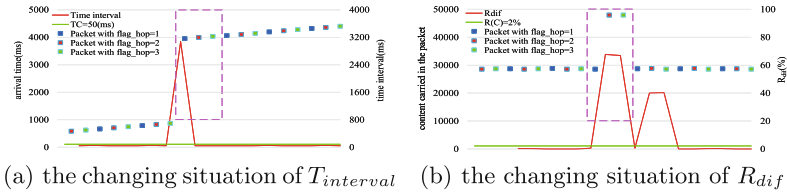


Fig. 7. Packet loss location detection mechanism in telemetry server

Runtime Reconfigurability. We conduct experiments to verify NT-RP has real-time task switching capability by sending new telemetry task commands during a telemetry process and observing whether the traffic throughput is affected. We consider three cases as 1) *MeasurementID* = 0 × 01 to measure flow one-way average latency, 2) *MeasurementID* = 0 × 02 to measure flow average throughput, 3) *MeasurementID* = 0 × 03 to measure link available bandwidth. Figure 8(a) shows that the traffic throughput hardly changes when the task command updated, indicating that the task switch does not have an additional impact on the system. The results in Fig. 8(b) show that the *MeasurementID* carried in the telemetry packet changes successfully according to the different commands. These two experiments confirm the runtime reconfigurability of NT-RP.

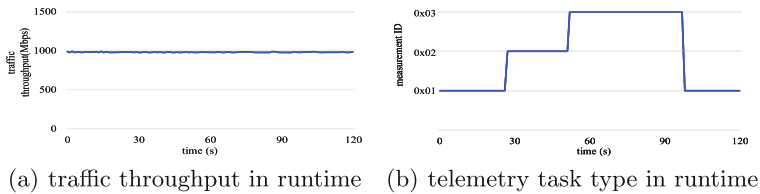


Fig. 8. Performance of NT-RP during task switching

5 Conclusion

In this paper, we propose a dynamically reconfigurable FPGA pipeline NT-RP, which can switch the calculation logic in runtime to obtain the measurements. It is experimentally demonstrated that NT-RP is well integrated with packet forwarding with almost no extra time load, and can accomplish accurate link-level network telemetry with less data overhead. Meanwhile, by finding the location of packet loss, NT-RP mitigate the telemetry information missing problem caused by packet loss. It is conceivable that by deploying NT-RP in scenarios such as

core switch clusters, network management and service traffic transmission can be parallelized more efficiently and robustly.

Acknowledgement. This work was supported by the National Key R&D Program of China (2018YFB1800602), China Postdoctoral Science Foundation (2022M710677), and Jiangsu Funding Program for Excellent Postdoctoral Talent.

References

1. Ben Basat, R., Ramanathan, S., Li, Y., Antichi, G., Yu, M., Mitzenmacher, M.: Pint: probabilistic in-band network telemetry. In: Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication, pp. 662–680 (2020)
2. Kim, Y., Suh, D., Pack, S.: Selective in-band network telemetry for overhead reduction. In: 2018 IEEE 7th International Conference on Cloud Networking (CloudNet), pp. 1–3. IEEE (2018)
3. Li, J., Yang, X., Sun, Z.: DrawerPipe: a reconfigurable packet processing pipeline for FPGA. *J. Comput. Res. Dev.* **55**(4), 717 (2018)
4. Min, L., Jinglin, S., Zhongcheng, L., Zhigang, K., Jian, M.: A new end-to-end measurement method for estimating available bandwidth. In: Proceedings of the Eighth IEEE Symposium on Computers and Communications, ISCC 2003, pp. 1393–1400. IEEE (2003)
5. Pezaros, D.P., Georgopoulos, K., Hutchison, D.: High-speed, in-band performance measurement instrumentation for next generation IP networks. *Comput. Netw.* **54**(18), 3246–3263 (2010)
6. Tan, L., et al.: In-band network telemetry: a survey. *Comput. Netw.* **186**, 107763 (2021)
7. Tan, L., Su, W., Zhang, W., Shi, H., Miao, J., Manzaneres-Lopez, P.: A packet loss monitoring system for in-band network telemetry: detection, localization, diagnosis and recovery. *IEEE Trans. Netw. Serv. Manag.* **18**(4), 4151–4168 (2021)
8. Tang, S., Li, D., Niu, B., Peng, J., Zhu, Z.: Sel-INT: a runtime-programmable selective in-band network telemetry system. *IEEE Trans. Netw. Serv. Manag.* **17**(2), 708–721 (2019)
9. Wang, T., Yang, X., Antichi, G., Sivaraman, A., Panda, A.: Isolation mechanisms for {High-Speed}{Packet-Processing} pipelines. In: 19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 2022), pp. 1289–1305 (2022)
10. Yang, X., et al.: Fast: enabling fast software/hardware prototype for network experimentation. In: 2019 IEEE/ACM 27th International Symposium on Quality of Service (IWQoS), pp. 1–10. IEEE (2019)
11. Zhao, Y., Cheng, G., Duan, Y., Gu, Z., Zhou, Y., Tang, L.: Secure IoT edge: threat situation awareness based on network traffic. *Comput. Netw.* **201**, 108525 (2021)



An Effective Comprehensive Trust Evaluation Model in WSNs

Chengxin Xu¹, Wenshuo Ma^{2(✉)}, and Xiaowu Liu¹

¹ School of Computer Science Qufu Normal University, Rizhao 276800, China

² School of Information Technology, Qingdao Vocational and Technical College of Hotel Management, Qingdao 266100, China

weimws@foxmail.com

Abstract. Wireless Sensor Networks (WSNs) are vulnerable to many security threats from compromised nodes. A trust management system is an effective method to detect the malicious behaviors in WSNs. In this paper, an effective Comprehensive Trust Evaluation Model (CTEM) for WSNs is proposed and two kinds of trusts are considered, the direct trust and the indirect trust. The direct trust is assessed by monitoring the data collection, the energy consumption and the data forwarding of node. More significantly, the entropy theory is introduced to measure the uncertainty of direct trust. The indirect trust is integrated to evaluate a comprehensive trust when the uncertainty of direct trust is high enough so as to improve the one-sidedness of direct trust. CTEM can not only reduce the computation overhead of node but also prolong the lifetime of network. Simulation results show that the proposed strategy can defend against internal attacks and have better performances compared with some typical trust evaluation mechanisms.

Keywords: Wireless sensor networks · Trust metrics · Trust evaluation system

1 Introduction

Wireless Sensor Networks (WSNs) are multi-hop and self-organized distributed networks consisting of hundreds even thousands of tiny sensor nodes. The low-cost, low-power, and resource-limited sensor nodes are usually used to collect, transfer and process the sensing data. This makes WSNs perform a variety of complex tasks and play an significant role in various applications [1], such as the battlefield surveillance, the smart cities, the medical surveillance and the emergency response. However, sensor nodes are easily exposed to the random failures and the cyber attacks due to the wireless characteristics and the deficiency in the secure fortification [2]. The confidentiality, the integrity and the availability

This work is supported by NSF of China under Grants 61672321; the Natural Science Foundation of Shandong Province under Grants ZR2021QF050 and ZR2021MF075); the Graduate Education Quality Courses of Shandong Province (SDYKC21097).

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022
L. Wang et al. (Eds.): WASA 2022, LNCS 13473, pp. 108–116, 2022.
https://doi.org/10.1007/978-3-031-19211-1_9

of the whole network may be compromised if a node is attacked. Therefore, it is a challenging work to detect malicious behaviors and relieve the negative effect of compromised nodes in a WSN. The trust management system is one of the preferred one to resist malicious attacks [3]. A successful trust model can provide useful information to identify a sensor node is trustworthy or not and encourages sensor nodes to act in a normal manner [4].

To address the aforementioned issues, this paper proposes an effective Comprehensive Trust Evaluation Model (CTEM) for WSNs. In CTEM, the trust metrics of node and the overall trust level are represented more rationally. The trust evaluation performance is improved through improving the correlation factor assignment scheme. Firstly, we divide a WSN into cluster topology. Then, the direct trust value is evaluated by three trust metrics of sensor nodes: data perception trust, energy trust and data forwarding trust. Secondly, the uncertainty level of direct trust is evaluated based on the entropy theory which is calculated in a novel way. If the uncertainty level of the direct trust values are high enough, the indirect trust values are assessed according to the neighbor node recommendation. Thirdly, we introduce a trust factor to trust evaluation with the aim of fusing the direct trust and the indirect trust to obtain a comprehensive trust of sensor nodes. Finally, CTEM is applied into the in-network data aggregation and nodes with high trustworthiness are selected as reliable relay node for data transmission, which avoids the risk of malicious aggregation node.

The remainder of this paper is organized as follows. Section 2 shows the related work. Section 3 describes the proposed comprehensive trust evaluation model in detail. And the simulation experiments are demonstrated in Sect. 4. Finally, we conclude this paper in Sect. 5.

2 Related Works

Various trust models, such as the entropy trust model, the fuzzy logic trust model, the D-S evidence trust model and the game theory trust model, are widely used in current literatures. Mathapati *et al.* [5] proposed a secure routing scheme with multi-dimensional trust evaluation. The proposed system adopted the analytical methodology to assess the multiple trust levels for sensors in the process of data aggregation. A Trust Management-based Secure Routing Scheme (TMSRS) was proposed for the industrial sensor networks with fog computing [6]. In TMSRS, a Gaussian distribution model was constructed to update the trusts according to the interaction among sensor nodes which provides a trade-off among security, energy and transmission performance. In [7], a secure cluster head election algorithm and a misbehavior detection mechanism were discussed. A node was elect as a cluster head according to the distance, the energy and the trust degree of node. The malicious nodes were detected and isolated from the network. Anwar *et al.* [8] proposed a Belief based Trust Evaluation Mechanism (BTEM). The Bayesian estimation was introduced into the trust evaluation based on the direct and indirect trusts. BTEM considers the correlation between the data gathered before and the estimated value of next collection round. However, this may negatively affect the robustness of trust assessment. In addition, the adaptability of pre-defined threshold is also a challenging issue.

3 Proposed Comprehensive Trust Assessment Model

3.1 Network Topology Model and Assumptions

A multi-level cluster topology is designed and an aggregation tree rooted at the Base Station (BS) is constructed as shown in Fig. 1. Nodes can be identified as the Cluster Head (CH) and the Cluster Member (CM). CMs can directly communicate with their CH which forwards the aggregation result to BS hop-by-hop. The Monitoring Node (MN) is responsible for monitoring communication channels and collecting global information. The sensor nodes are organized into clusters with the existing clustering schemes. In a cluster, CH and MN work together to record and update the trust value of CMs. And all sensor nodes are trustworthy and the BS is secure enough in the initial stage.

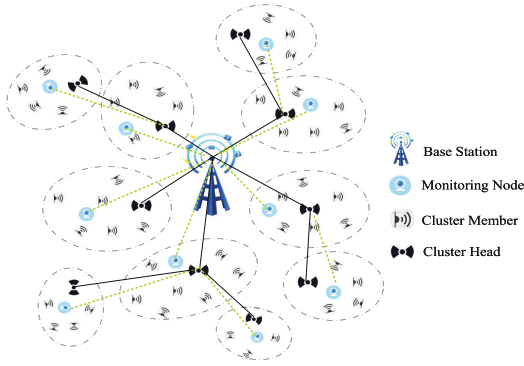


Fig. 1. Network topology

3.2 Direct Trust Model

Data Perception Trust. It is a regular requirement for CH to evaluate the Data Perception Trust (DPT) of sensor nodes in order to ensure the reliability of data collection. DPT reflects the data quality and consistency when there are error and uncertainty in sensing data. Without loss of generality, assumed that B is a CM with a sending data s_B and S is the set of sensing data received from the neighbors of B . e_i is one of sensor data in S ($e_i \in S$) and the average of sensing data in S is ξ . In the n th sensing round, the data error of node B is expressed as $\Delta d_B(n) = |s_B - \xi|$. The data perception trust of node B in CH (represented as A) is $DPT_A^B(n)$ ($DPT_A^B(n) \in [0, 1]$). If $\Delta d_B(n) < v$, we have

$$DPT_A^B(n) = DPT_A^B(n-1) + \frac{1 - DPT_A^B(n-1)}{\rho}, \quad (1)$$

otherwise,

$$DPT_A^B(n) = DPT_A^B(n-1) - \frac{DPT_A^B(n-1)}{\sigma}. \quad (2)$$

where $DPT_A^B(n-1)$ denotes DPT of node B evaluated by node A at sensing round $(n-1)$. v is the tolerable error. The parameters ρ and σ determine the increasing or decreasing rate of DPT.

Energy Trust. The Energy Trust (ET) can be used to identify whether a suspicious node launching a malicious attack to exhaust the energy and harm the security of sensor node. We design two metrics to measure ET. One is the residual energy ratio which is the ratio of the residual energy to the initial energy. The residual energy ratio of CM at sensing round t is re_t after a CM sends its data to CH. We assume that the residual energy ratio threshold is th_{re} . A CM is allowed to participate in the data collection when its residual energy ratio is greater than th_{re} . The other is the energy consumption rate. We use energy consumption rate to detect the abnormal energy changing of nodes and the energy consumption rate can be illustrated as

$$\Delta p = \frac{|p_t - p_{t-1}|}{p_{t-1}}. \quad (3)$$

If Δp ($\Delta p \in [0, 1]$) exceeds the a threshold $th_{\Delta p}$, the CH determines that the CM is abnormal and sets the energy trust to a lower level even zero. Based on these indexes, the energy trust of node A evaluated by node B at round t can be expressed as

$$ET_A^B(t) = \begin{cases} re_t(1 - \Delta p), & re_t > th_{re} \ \& \ \Delta p < th_{\Delta p} \\ 0, & re_t < th_{re} \ | \ \Delta p > th_{\Delta p} \end{cases}, \quad (4)$$

Data Forwarding Trust. The Data Forwarding Trust (DFT) is used to evaluate the reliability of nodes' forwarding and communication behaviors. Node A monitors the behaviors of node B and counts the number of packets received and forwarded by node B . The forwarding ratio of node B , $FR_A^B(t) = \frac{p_t}{(q_t + p_t)}$, where p_t and q_t denote the number of packets forwarded and discarded by node B at round t , respectively. The forwarding ratio can imply the packet forwarding behavior of node B and the consistency of node forwarding behavior can be assessed by comparing $FR_A^B(t)$ at two consecutive rounds as following three cases.

- i) If $FR_A^B(t) > FR_A^B(t-1)$, the δ_t increases and $\delta_t = \delta_{t-1} - \alpha(FR_A^B(t) - FR_A^B(t-1))$.
- ii) If $FR_A^B(t) < FR_A^B(t-1)$, the δ_t decreases and $\delta_t = \delta_{t-1} - \beta(FR_A^B(t) - FR_A^B(t-1))$.
- iii) if $FR_A^B(t) = FR_A^B(t-1)$, $\delta_t = \delta_{t-1}$.

In the above cases, δ_t ($\delta_t \in [0, 1]$) indicates the abnormal fluctuation of forwarding behavior. α is the penalty factor and β is the reward factor. The α should be greater than β , which means that the discarded behavior (bad behavior) has more influences on the forwarding ratio than the forwarding behavior

(good behavior). This mechanism makes the malicious node lose DFT quickly when bad behavior occurs. Meanwhile, it requires a long time or a lot of normal behaviors to recover its DFT to a higher level. Then, the DFT of node B evaluated by node A can be expressed as

$$DFT_A^B(t) = FR_A^B(t) * \cos(\delta_t * \frac{\pi}{2}). \tag{5}$$

Direct Trust. Based on the above analysis, the Direct Trust (DT) of sensor node can be formalized depending on the DPT, the ET and the DFT. We describe the DT of node B as

$$\begin{cases} DT_A^B = \mu_1 DPT_A^B + \mu_2 ET_A^B + \mu_3 DFT_A^B \\ s.t. \min\{DPT_A^B, ET_A^B, DFT_A^B\} \geq th_{DT} \end{cases}, \tag{6}$$

where $DT_A^B \in [0, 1]$. μ_1, μ_2 and μ_3 represent the weights of three trust metrics in the evaluation of DT and satisfy $\sum_{i=1}^3 \mu_i = 1$. The parameter th_{DT} are used to determine whether the three indexes are reliable or not. If one or more trusts were less than th_{DT} , the node will be directly identified as a malicious node.

3.3 Indirect Trust Model

Besides the DT, the Indirect Trust (IT) also needs to be evaluated. In the communication radius of sensor nodes, we select a set of common neighbors between source and destination. These common neighbors provide the recommendation trust of destination node to the source node. The recommendation trust can be illustrated using Fig. 2.

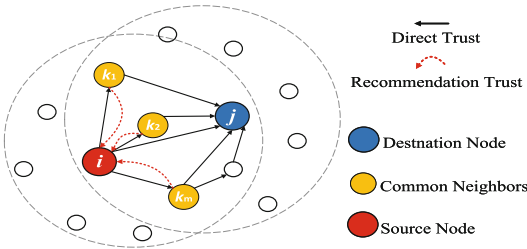


Fig. 2. Trust evolution of sensor nodes

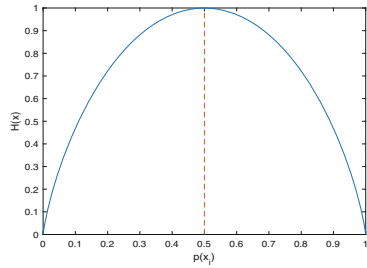


Fig. 3. Accuracy of the proposed model

In Fig. 2, node i and node j are the source node and the destination node, respectively. The distance between i and j is less than the communication radius. We select a set of common neighbors of i and j , $k_1, k_2, k_3, \dots, k_m$ and the direct trusts in node i for k common neighbors are larger than the trust threshold, namely node i considers these neighbor nodes are trustworthy. During

the trust evaluation process, node i sends query messages to these k common neighbors which send back the recommendation trusts of destination node j . These recommendation trusts serve as the basis for node i to evaluate the IT of node j . Then, the IT of node j in node i can be expressed as

$$IT_j^i(t) = \sum_{k \in S_f} w_k * RT_{i,j}^k(t), \quad (7)$$

where S_f is the set of common neighbors of nodes i and node j . w_k is the weight of common neighbor nodes k .

$$w_k = \frac{DT_A^k(t)}{\sum_{k=1}^m DT_A^k(t)}, \sum_{k \in S_f} w_k = 1 (0 \leq w_k \leq 1) \quad (8)$$

$RT_{i,j}^k(t)$ is the recommended trust of node k for node j . Due to the transitivity of trust, the recommendation trust of common neighbor nodes for node j , $RT_{i,j}^k(t)$, can be formalized as follows.

$$RT_{i,j}^k(t) = \frac{\sum_{k=0}^m DT_i^k(t) * DT_k^j(t)}{m} \quad (9)$$

$DT_i^k(t)$ is the DT in node i for node k , $DT_k^j(t)$ is the DT in node k for node j , and m is the number of common neighbors of nodes i and node j .

3.4 Comprehensive Trust Evaluation Model

The proposed model uses entropy theory to evaluate the comprehensive trust. Usually, entropy theory is used to measure the uncertainty in a random event system. The uncertainty function F is a monotonic decreasing function of probability P , which is shown as $F(P) = \log \frac{1}{P} = -\log P$.

A random event contains many possible cases, each of which has its own uncertainty. The amount of information of this event is the sum of the uncertainties of all cases' occurrences. Let an event has n cases with the values $U_1, U_2 \dots U_n$ and happens in a information source with the probabilities $P_1, P_2 \dots P_n$, respectively. n values are independent with each other and the average uncertainty of the information source (information entropy) can be expressed as

$$H(U) = E(F(P_i)) = - \sum_{i=1}^n P_i \log_2 P_i, \quad (10)$$

where U represents the set of all possible event generated by information source. The entropy function image can be illustrated by Fig. 3. The curve shows that the more chaotic the system has, the greater the entropy is.

We evaluate the uncertainty of DT by calculating its entropy. The DT of every CM in a cluster need to be recorded in each sensing round. Let there be m CMs in a cluster and n DTs for each CM after the n -th sensing rounds. The DT of a node is denoted by a vector x , the matrix X of all CMs in a cluster is:

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix}. \quad (11)$$

It indicates that a node may be a suspicious node one if its x_{ij} fluctuates greatly. Then, we will discuss the uncertainty measurement of DT by its information entropy. It takes three steps

i) Normalize the DT matrix:

$$x'_{ij} = \frac{x_{ij} - \min\{x_j\}}{\max\{x_j\} - \min\{x_j\}}. \quad (12)$$

ii) Compute the weight $p(x_{ij})$ of DT after the j -th sensing round of the node i :

$$p(x_{ij}) = \frac{x'_{ij}}{\sum_{i=1}^m x'_{ij}}. \quad (13)$$

iii) Calculate the DT entropy $H(x)$ of node i :

$$H(x) = -k \sum_{i=1}^m p(x_{ij}) \ln p(x_{ij}), \quad (14)$$

where $k = 1/\ln m$, $k > 0$, $0 \leq H(x) \leq 1$.

In (14), $H(x)$ is the DT entropy of a CM (short for B , A for CH), that is $H(DT_A^B)$. Let th_E be the threshold for uncertainty. When $th_E < H(DT_A^B) \leq 1$, it means that the uncertainty of direct trust is high and more relevant information is needed in order to accurately judge whether a node is malicious or not and the IT should be introduced. The CT_A^B can be expressed as:

$$CT_A^B(t) = \begin{cases} DT_A^B(t), & 0 < H(DT_A^B) \leq th_E \\ (1 - \mu) * DT_A^B(t) + \mu * IT_A^B(t), & th_E < H(DT_A^B) \leq 1 \end{cases}, \quad (15)$$

In (15), μ is the confidence factor which expresses the weight of the DT in the CT. The confidence factor can be expressed as $\mu = 1 - \eta^{-z}$, where η denotes the number of direct interactions between the sensor nodes. z is a dynamic value $z \in [0, 1]$ which is related to the actual application of WSNs. The number of direct interactions between A and B increases as the time goes on and the proportion of IT in CT also increases. This enhances the security of network.

4 Simulation Experiment and Performance Analysis

In this section, the performance of CTEM is verified using MATLAB R2020b. Several parameters of simulation scenarios are determined after combining various types of malicious attacks. CTEM is compared with other trust-based data

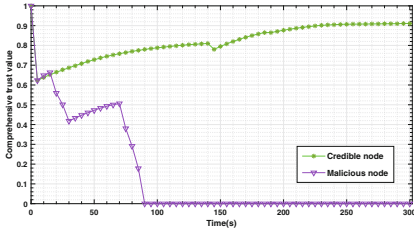


Fig. 4. Trust evolution of sensor nodes

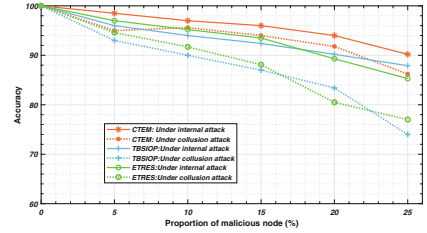


Fig. 5. Accuracy of the proposed model

aggregation models, ETRES [9] and TBSIOP [10], in terms of algorithm accuracy, communication overhead and the network lifetime.

The average of CT of malicious nodes and trusted nodes are shown in Fig. 4 within a specified time. When the entropy of DT is under the threshold, the CT is equal to the DT. As the time goes on, the entropy of DT increases and exceeds the threshold. The IT is participated in the process of trust evaluation and the CT fluctuates slightly. As the purple curve shows, the trust value of malicious node decreases rapidly when the malicious node launches the malicious attack. From this simulation experiment, CTEM can reflect the trust evolution according to the behavior of sensor nodes. CTEM demonstrates the highest accuracy compared with other mechanisms under the internal attacks and the collusion attack in Fig. 5.

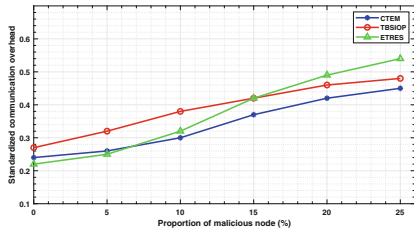


Fig. 6. Communication overhead

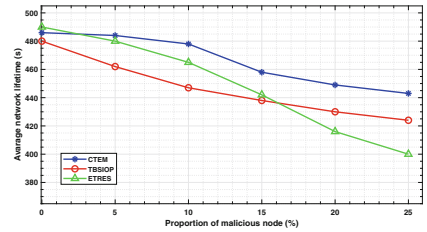


Fig. 7. Network lifetime

The normalized communication overhead represents the ratio of the number of control packets and the overall data packets. As shown in Fig. 6, CTEM has lower communication overhead than other algorithms, which performs better among the three. TBSIOP and CTEM demonstrates more consistently in communication overhead when the number of malicious nodes increases. In Fig. 7, we compare the network lifetime of CTEM with TBSIOP and ETRES. We can know that ETRES significant decreases the network lifetime when the number of malicious nodes reaches 15%. When the collusion attack occurs in the network, the lifetime of TBSIOP rapidly deteriorates and CTEM performed better. In brief, CTEM shows well performances in terms of accuracy, communication overhead and network lifetime.

5 Conclusion

A successful trust management system is an effective mechanism to ensure the security of WSNs and distinguish the malicious nodes from normal nodes. In this paper, we proposed a comprehensive trust evaluation model based on the behavior attributes and the interaction between sensor nodes. The trust is evaluated through the flexible and effective fusion of the direct trust and the indirect trust with the entropy theory. Theoretical analysis and simulation results prove that our proposed model can accurately evaluate the trust of sensor nodes at the low price of communication overhead and energy consumption. In the future, we will apply our model into more complex attack scenarios and discuss new trust models which integrate with other entropy theories.

References

1. Kassab, W., Darabkh, K.A.: A-Z survey of internet of things: architectures, protocols, applications, recent advances, future directions and recommendations. *J. Netw. Comput. Appl.* **163**, 102663 (2020)
2. Nguyen, T.-D., Zalyubovskiy, V., Le, D.-T., Choo, H.: Break-and-join tree construction for latency-aware data aggregation in wireless sensor networks. *Wirel. Netw.* **26**(7), 5255–5269 (2020). <https://doi.org/10.1007/s11276-020-02389-x>
3. Hajar, M.S., Al-Kadri, M.O., Kalutarage, H.K.: A survey on wireless body area networks: architecture, security challenges and research opportunities. *Comput. Secur.* **104**, 102211 (2021)
4. Fan, N., Wu, C.Q.: On trust models for communication security in vehicular ad-hoc networks. *Ad Hoc Netw.* **90**, 101740 (2019)
5. Mathapati, M., Kumaran, T.S., Muruganandham, A., Mathivanan, M.: Secure routing scheme with multi-dimensional trust evaluation for wireless sensor network. *J. Ambient Intell. Humaniz. Comput.* **12**(6), 6047–6055 (2020). <https://doi.org/10.1007/s12652-020-02169-7>
6. Fang, W., Zhang, W., Chen, W., Liu, Y., Tang, C.: TMSRS: trust management-based secure routing scheme in industrial wireless sensor network with fog computing. *Wirel. Netw.* **26**(5), 3169–3182 (2020)
7. Saidi, A., Benahmed, K., Seddiki, N.: Secure cluster head election algorithm and misbehavior detection approach based on trust management technique for clustered wireless sensor networks. *Ad Hoc Netw.* **106**, 102215 (2020)
8. Anwar, R.W., Zainal, A., Outay, F., Yasar, A., Iqbal, S.: BTEM: belief based trust evaluation mechanism for wireless sensor networks. *Future Gener. Comput. Syst.* **96**, 605–616 (2019)
9. Zhao, J., Huang, J., Xiong, N.: An effective exponential-based trust and reputation evaluation system in wireless sensor networks. *IEEE Access* **7**, 33859–33869 (2019)
10. Bangotra, D.K., Singh, Y., Selwal, A., Kumar, N., Singh, P.K.: A trust based secure intelligent opportunistic routing protocol for wireless sensor networks. *Wirel. Pers. Commun.* 1–22 (2021)



Precise Code Clone Detection with Architecture of Abstract Syntax Trees

Xin Guo¹, Ruyun Zhang², Lu Zhou¹, and Xiaozhen Lu¹(✉)

¹ Nanjing University of Aeronautics and Astronautics, Nanjing, China
{xin.guo, lu.zhou, luxiaozhen}@nuaa.edu.cn

² Zhejiang Lab, Hangzhou, China
zhangry@zhejianglab.com

Abstract. In the field of code clone detection, there are token-based similarity and abstract syntax tree-based detection methods. The former consumes less resources and is faster to detect, while the latter consumes more space and is less efficient. And there are few tools that scale to large-scale databases. To address the challenges, an approach is proposed that can detect code clones using the similarity of tokens and architecture of abstract syntax trees. Architecture of the syntax trees preserves the precision of detecting clone pairs, at the same time, the method also preserves the speed of matching code similarity. In the approach, it first parses the tokens of the code fragments and gets the features of the syntax trees. It can eliminate the unqualified parts of them based on the architecture when matching the candidates quickly, and then detects the similarity in detail. Finally, the results are output according to the input threshold range. The experiments confirm that the method substantially improves the precision of code clone detection while keeping the recall rate unabated.

Keywords: Code clone detection · Abstract syntax tree · Large-scale database

1 Introduction

Code clone refers to two or more identical or similar code fragments. When users reuse code, such as copying and pasting existing code fragments, code clone will be generated [3]. To a certain extent, code clone helps develop software systems and generates positive benefits [1]. Developers can improve efficiency through code clone when they use development frameworks or reuse design patterns [3]. However, long-term large-scale code clone leads to an ever-expanding code base and increased maintenance costs [1]. If there are bugs in the code, it also reduces the reliability of the software due to the constant propagation of code clone. Therefore, code clone detection is necessary.

Many clone detection tools based on text. SDD [7] and NiCad [13] have no code hierarchy, so they are inefficient and easy to lose information. CCAAligner [19], CCLearner [8] and so on are vocabulary-based code representations. They

have improved the utilization of the source code, but ignore the structural information of the source code. Then there are syntax-based clone detection technologies, such as CDLH [20], Deckard [4] and so on. But they are expensive and the efficiency is not enough. There are also many semantic-based code clone detection technologies, including Oreo [15], Duplix [6], GPLAG [10] and so on. But they are greatly expensive and difficult to deploy.

Most of the theories of clone detection are however focused on performance. Few tools can scale to the demands of clone detection in large code bases [14, 17]. Large-scale clone detection allows researchers to study code clone in open source development communities such as Git-Hub.

Tokens and syntax trees are currently the most common objects for determining code clone. Tokens have been utilised for the quantification of the similarity between code blocks. It consumes fewer resources and detects fast. But the precision is low with many false positives. Syntax trees represent the syntactic structure and hierarchy of the code. Matching syntax trees is a well-established approach. But it is inefficient and consumes large resources.

To explore effective methods for addressing such challenges, an approach is made to combine tokens with the architecture of abstract syntax trees. Intuitively, according to the definitions of code clone type 1 and type 2, they do not change the syntactic structure of the code blocks. This study has identified that code clone type 1 and 2 have the same architecture of abstract syntax trees, but this feature seems to have been overlooked by researchers. The approach therefore replaces the full syntax trees with the architectural features of them and uses them to filter the candidates in similarity determination. The advantage allows us to win-win with maximum efficiency and precision.

In the proposed method, the tokens of the code blocks and the architecture of syntax trees are first parsed. Then the tokens are sorted according to the frequency of all tokens. Sub-block overlap filtering is performed on the code blocks according to the partial index to get the candidates. Among the candidates, if the architecture of syntax trees are not uniform, the candidates are excluded. Finally, the final similarity is calculated and listed as the final results if they are within the threshold value.

The experimental results show that the approach can effectively eliminate the misclassification in type 2 within different thresholds, which leads to a great improvement in precision. The experimental data show that the threshold value corresponding to type 1 is 100%, and the best corresponding ranges for type 2 are [90%, 100%). In addition, the method improves the detection efficiency. The larger the data size is, the more obvious the efficiency improvement is.

This work makes the following contributions:

- An precise approach is proposed to detect code clone with combining tokens with the architecture of abstract syntax trees.
- Precision of clone type 2 is significantly improved.
- It concluded that the best similarity for clone type 1 is 100% and for clone type 2 is [90%, 100%).
- Time of detection part is reduced corresponding to different threshold ranges.

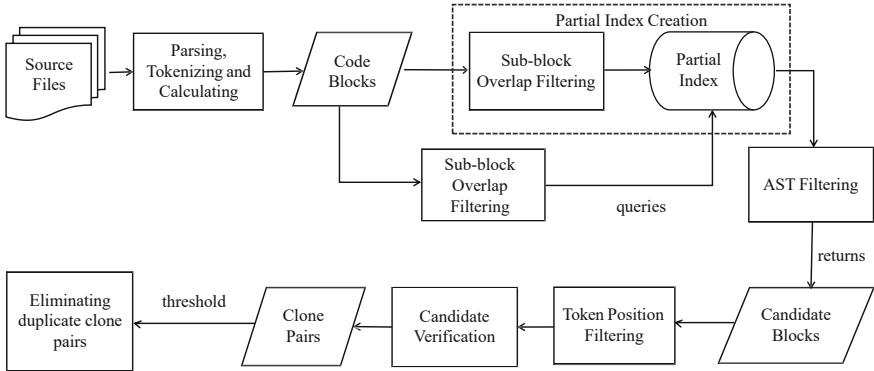


Fig. 1. Introduce abstract syntax trees' clone detection process.

2 Background

Code Clone. Researchers uniformly use the following basic concept definitions [2]. (1) **Type 1** (exactly the same code): Except for white spaces and comments, two code fragments are exactly the same. (2) **Type 2** (renamed code): Two code fragments are the same except for the corresponding names of user-defined identifiers, types, layout and comments. (3) **Type 3** (almost the same code): Syntactically similar fragments that differ at the statement level. (4) **Type 4** (semantic similar code): Heterogeneous codes with the same function are not similar in text or syntax, but have similarities in semantics.

Abstract Syntax Tree. An abstract syntax tree (AST), or syntax tree for short, is an abstract representation of the grammatical structure of source code.

3 Methodology

According to the definition of code clone, type 1 and 2 do not change the structure of code, while type 3 does. Focusing on tokens only will lose structural information, while focusing on the syntax trees' structure only, the compensation is large and inefficient. In order to rectify the problem of clone detection in large-scale database, the approach is given in combining tokens and AST. The advantage is that it replaces the complete AST with structural features. To illuminate this uncharted area, SourcererCC was selected to join experimental procedure [16]. The advantage of SourcererCC is that it's currently one of the most efficient and accurate methods in detecting code clone of large-scale databases. But the problem is that there is room to improve the precision.

Figure 1 illustrates the exact procedure of the approach. First, it specifies the upper and lower limits of similarity. In parsing and tokenizing, it parses out the tokens and AST of the query blocks and candidate blocks. Notably, it preserves the height and width of AST instead of the whole AST. Next, all code block tokens are sorted according to the frequency of them. Partial indexes are

calculated based on the threshold and candidates are selected by whether the sub-blocks overlap. If query blocks have different features with the candidates, they will be eliminated from the candidates. Iterating the tokens of query and candidate blocks can calculate the corresponding similarity. The detection process is performed twice with the lower and upper limit of the threshold range and duplicate clone pairs are removed.

3.1 Parsing and Calculating

It's necessary to save token information and structural features of the query blocks and candidate blocks respectively. The method parses the tokens and AST of functions. The tokens need to be statistically global frequency map, and these functions sort their tokens in frequency order. To extract the structural information of AST, the height and width are generic features.

3.2 Abstract Syntax Tree Filtering

According to the definition of clone types. The major difference between type 1, 2 and 3, 4 is the change in structure. So if the structure features of one code block are different from the other, it means that they must not be code clone type 1 or 2. The height and width are selected as features of AST in the approach.

In this section, based on the partial index of the query blocks, determining whether sub-blocks overlap can eliminate most of the non-candidates. Then, among the remaining candidates, it leaves the candidates which have the same height and width with query blocks. This eliminates most of the misclassification by the trees' structure.

3.3 Eliminate Duplicate Clone Pairs

SourcererCC outputs clone pairs whose similarity is greater than or equal to the set threshold. For example, if the threshold is set to 80%, clone pairs with a threshold of 100% (or Type 1) are also included. If the user only needs the results of clone type 2, the output should clear the data for which the threshold is set to 100%. Thus the method needs to exclude data of different clone types according to the threshold.

Formally, the user wants to get the clone pair $\theta_1 \sim \theta_2$ about the threshold range. It then runs the detection process twice, including sub-block overlap filtering, syntax tree architecture filtering and determining similarity. Finally the experimental results with θ_1 are removed from the experimental results with θ_2 .

4 Experimental Setup

4.1 Research Questions

- **RQ-1.** Does the architecture of AST play a role in improving recall rate?
- **RQ-2.** Does the architecture of AST play a role in improving precision rate?
- **RQ-3.** Whether there is a relatively suitable range of similarity thresholds for type 1 and type 2?
- **RQ-4.** How does architecture of AST affect program execution time?

Table 1. The number of detected clone pairs.

Method	100%	[90%, 100%)	[80%, 100%)	[70%, 100%)
SourcererCC	958,777	532,450	1,097,642	2,128,502
Our method	940,710↓	367,581↓	546,757↓	678,707↓

4.2 Dataset and Configuration

The experimental data is provided by IJaDataset in BigCloneBench [18]. The programming language is Java. To see the results more intuitively, we divided data set into 830,684 functions with a total size of 531.9 MB.

To observe the distinction between different threshold ranges, four test processes are conducted, including 100%, [90%, 100%), [80%, 100%) and [70%, 100%). The experiment uses 20 tokens as the minimum, 50,000 as the maximum.

5 Experimental Results and Analyses

The results prove that the approach is feasible and produces good effect. The AST can eliminate many erroneous clone pairs, making the precision much higher. According to IJaDataset, the number of detected clone pairs is shown in Table 1. As the range expands, the amount of clone pairs increases a lot. While our approach excludes a large number of clone pairs in each ranges. The larger the threshold range is, the more cases are excluded.

Either the BigCloneBench or the random sampling method, our method is twice as precise as SourcererCC. While there is no change in recall rate. The threshold value for type 1 is 100% and the most suitable threshold range for type 2 is [90%, 100%). And in terms of efficiency, the larger the amount of data, the more the detection time saved by our method.

5.1 Recall

The confusion matrix is a standard format for precision evaluation. For a two-class classification system, the pattern classifier has four classification results: (1) **TP** (True Positive): a correct positive example, an instance is positive and is also judged as positive. (2) **FN** (False Negative): a false negative example, an instance is positive but judged as negative. (3) **FP** (False Positive): a false positive example, an instance is negative but judged as positive. (4) **TN** (True Negative): a correct negative example, an instance is negative and is also judged negative.

$$Recall = \frac{TP}{TP + FN} \quad (1)$$

The recall rate refers to the proportion of the correct prediction that is positive to all that is actually positive. Table 2 shows the evaluation metrics of different thresholds. The recall in our method is the same as SourcererCC. However, AST

Table 2. Recall and precision in BigCloneBench.

Similarity threshold	Experimental values	Method		
		SourcererCC	Our method	
100%	Confusion	TP	48,112	48,112
		FP	1,036	899
	Matrix	TN	8,535,001	8,535,138
		FN	4	4
	Recall/%	99.99	99.99	
	Precision/%	97.89	98.17↑	
[90%, 100%)	Confusion	TP	3,573	3,573
		FP	5,622	3,503
	Matrix	TN	8,574,297	8,576,416
		FN	661	661
	Recall/%	84.39	84.39	
	Precision/%	38.86	50.49↑	

Similarity threshold	Experimental values	Method		
		SourcererCC	Our method	
[80%, 100%)	Confusion	TP	3,967	3,967
		FP	11,071	4,664
	Matrix	TN	8,568,848	8,575,255
		FN	267	267
	Recall/%	93.69	93.69	
	Precision/%	26.38	45.96↑	
[70%, 100%)	Confusion	TP	4,156	4,156
		FP	21,981	5,774
	Matrix	TN	8,557,938	8,574,145
		FN	78	78
	Recall/%	98.18	98.18	
	Precision/%	15.90	41.85↑	

Table 3. Precision by random sampling

Method	100%	[90%, 100%)	[80%, 100%)	[70%, 100%)
SourcererCC/%	98.67	37.33	32.66	25.33
Our method/%	98.67	56.67↑	54.00↑	51.33↑

clone detection has fewer FP and more TN, indicating that it identifies clone pairs that were originally incorrectly determined to be clone type 2. It confirms that the architecture of AST can distinguish most of type 2 and type 3.

⚡ **RQ-1:** ▶ *The recall of AST clone detection is the same as SourcererCC. While the method generates fewer clone pairs and consumes fewer memory resources.* ◀

5.2 Precision

Precision refers to the proportion of correct predictions that are positive to all predictions that are positive. In the BigCloneBench, the statistical precision is shown in Table 2.

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

It's not difficult to find that the method has the same TP as SourcererCC, while it has fewer FP. It means there are fewer cases of misclassification. In [90%, 100%) to [70%, 100%), the precision improves by 11.6%, 19.6% and 26%, respectively. It shows that the precision has been greatly improved.

In order to estimate the precision of the tools better, scholars manually validate a random sample of their output [16]. 150 clone pairs were randomly selected as samples from each corresponding result. The precision is shown in Table 3. Except for 100%, the precision of SourcererCC is around 30%, while our method improves it to more than 50%. The results support that AST can help providing a significant improvement in the precision of code clone detection.

Table 4. Time comparison

Method	Parsing/s	Detecting/s	Similarity threshold
SourcererCC	130.94	109.06	100%
		149.10	[90%, 100%)
		370.96	[80%, 100%)
		1561.21	[70%, 100%)
Our method	1113.76↑	111.36↑	100%
		144.58↓	[90%, 100%)
		313.95↓	[80%, 100%)
		940.71↓	[70%, 100%)

⚡ **RQ-2:** ► *Whether in the BigCloneBench database or random sample detection, the precision is significantly improved.* ◀

5.3 Optimal Range

Combining Table 2 and 3, 99.99% of type 1 is distributed in the range of threshold value 100%. As for type2, the precision is highest in [90%, 100%). Recall is over 80%. There are more FPs than TPs in [80%, 100%), indicating that type 3 clones are more suitable for this threshold range.

⚡ **RQ-3:** ► *Clone type 1 corresponds to the most suitable threshold value of 100%, and type 2 is [90%, 100%).* ◀

5.4 Execution Time

In Table 4, the code detection divides the whole process into two parts, one is parsing and the other is detecting. The AST detection method requires more parsing time, since parsing the AST takes most of the time. In actual situations, the database is fixed, and the intermediate file can be parsed in advance.

When the threshold is 100%, the method adds 2s. However, the time is reduced in type 2 detection. The lower the lower limit of the threshold range, the more time is saved. Because in the case of more orders of magnitude, the AST filtering deletes a large number of candidates.

⚡ **RQ-4:** ► *The method requires more parsing time, but reduces the detection time. The scale of database is larger, the more time can be saved.* ◀

6 Discussion

Through the experimental results, the AST method can effectively eliminate the misjudged clone pairs in type 2. However, many type 3 examples' architectures

are the same, which means that although the structure of the tree has changed, the height and width are not affected. Perhaps it should find more effective features that can continue to improve the precision in future studies.

7 Related Work

Many clone detection tools based on text. For example, NiCad [13] and SDD [7] and so on have no code hierarchy, so they are inefficient and easy to lose information. CCFinder [5], CP-Miner [9], CCAaligner [19], CCLearner [8], Boreas [21], FRISC [11] and CDSW [12] are vocabulary-based code representations. They have improved the utilization of the source code, but ignore the structural information of the source code. At the same time, this kind of method is sensitive to code statement modification, and it is easy to miss code clones with only specific nuances [3]. Then there are syntax-based clone detection technologies, such as Deckard [4], CDLH [20] and so on. The advantage is that it can take into account the structural characteristics of the source code, and the disadvantage is that the difference between identifiers and text values cannot be recognized. There are also many semantic-based code clone detection technologies, including Duplix [6], GPLAG [10], Oreo [15] and so on. But they are greatly expensive, steps are complicated and difficult to deploy.

8 Conclusion

The first main contribution proposed in this field is a precise code clone detection. It improves the precision greatly and reduces the detecting time. In addition, it can output the results of the specified threshold range. This study finds that type 1 corresponds to 100%, and type 2 corresponds to [90%, 100%) more appropriately.

Acknowledgements. This work was supported by the National Key R&D Program of China (No. 2020AAA0107704).

References

1. Aversano, L., Cerulo, L., Di Penta, M.: How clones are maintained: an empirical study. In: 11th European Conference on Software Maintenance and Reengineering (CSMR 2007), pp. 81–90. IEEE (2007)
2. Bellon, S., Koschke, R., Antoniol, G., Krinke, J., Merlo, E.: Comparison and evaluation of clone detection tools. *IEEE Trans. Software Eng.* **33**(9), 577–591 (2007)
3. Chen, Q.Y., Shan-Ping, L.I., Yan, M., Xia, X.: Code clone detection: a literature review. *J. Softw.* (2019)
4. Jiang, L., Misherghi, G., Su, Z., Glondu, S.: DECKARD: scalable and accurate tree-based detection of code clones. In: 29th International Conference on Software Engineering (ICSE 2007), pp. 96–105. IEEE (2007)
5. Kamiya, T., Kusumoto, S., Inoue, K.: CCFinder: a multilinguistic token-based code clone detection system for large scale source code. *IEEE Trans. Software Eng.* **28**(7), 654–670 (2002)

6. Krinke, J.: Identifying similar code with program dependence graphs. In: Proceedings Eighth Working Conference on Reverse Engineering, pp. 301–309. IEEE (2001)
7. Lee, S., Jeong, I.: SDD: high performance code clone detection system for large scale source code. In: Companion to the 20th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, pp. 140–141 (2005)
8. Li, L., Feng, H., Zhuang, W., Meng, N., Ryder, B.: CCLearner: a deep learning-based clone detection approach. In: 2017 IEEE International Conference on Software Maintenance and Evolution (ICSME), pp. 249–260. IEEE (2017)
9. Li, Z., Shan, L., Myagmar, S., Zhou, Y.: CP-Miner: finding copy-paste and related bugs in large-scale software code. *IEEE Trans. Software Eng.* **32**(3), 176–192 (2006)
10. Liu, C., Chen, C., Han, J., Yu, P.S.: GPLAG: detection of software plagiarism by program dependence graph analysis. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 872–881 (2006)
11. Murakami, H., Hotta, K., Higo, Y., Igaki, H., Kusumoto, S.: Folding repeated instructions for improving token-based code clone detection. In: 2012 IEEE 12th International Working Conference on Source Code Analysis and Manipulation, pp. 64–73. IEEE (2012)
12. Murakami, H., Hotta, K., Higo, Y., Igaki, H., Kusumoto, S.: Gapped code clone detection with lightweight source code analysis. In: 2013 21st International Conference on Program Comprehension (ICPC), pp. 93–102. IEEE (2013)
13. Roy, C.K., Cordy, J.R.: NICAD: accurate detection of near-miss intentional clones using flexible pretty-printing and code normalization. In: 2008 16th IEEE International Conference on Program Comprehension, pp. 172–181. IEEE (2008)
14. Roy, C.K., Zibrán, M.F., Koschke, R.: The vision of software clone management: past, present, and future (keynote paper). In: 2014 Software Evolution Week-IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE), pp. 18–33. IEEE (2014)
15. Saini, V., Farmahinifarahani, F., Lu, Y., Baldi, P., Lopes, C.V.: Oreo: detection of clones in the twilight zone. In: Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pp. 354–365 (2018)
16. Sajnani, H., Saini, V., Svajlenko, J., Roy, C.K., Lopes, C.V.: SourcererCC: scaling code clone detection to big-code. In: Proceedings of the 38th International Conference on Software Engineering, pp. 1157–1168 (2016)
17. Svajlenko, J., Keivanloo, I., Roy, C.K.: Scaling classical clone detection tools for ultra-large datasets: an exploratory study. In: 2013 7th International Workshop on Software Clones (IWSC), pp. 16–22. IEEE (2013)
18. Svajlenko, J., Roy, C.K.: Evaluating clone detection tools with BigCloneBench. In: 2015 IEEE International Conference on Software Maintenance and Evolution (ICSME), pp. 131–140. IEEE (2015)
19. Wang, P., Svajlenko, J., Wu, Y., Xu, Y., Roy, C.K.: CCAliGner: a token based large-gap clone detector. In: Proceedings of the 40th International Conference on Software Engineering, pp. 1066–1077 (2018)

20. Wei, H., Li, M.: Supervised deep features for software functional clone detection by exploiting lexical and syntactical information in source code. In: IJCAI, pp. 3034–3040 (2017)
21. Yuan, Y., Guo, Y.: Boreas: an accurate and scalable token-based approach to code clone detection. In: Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering, pp. 286–289 (2012)



Multi-view Pre-trained Model for Code Vulnerability Identification

Xuxiang Jiang¹, Yin hao Xiao², Jun Wang¹, and Wei Zhang¹(✉)

¹ School of Computer Science and Technology, East China Normal University, Shanghai, China

zhangwei.thu2011@gmail.com

² School of Information Science, Guangdong University of Finance and Economics, Guangzhou, China

Abstract. Vulnerability identification is crucial for cyber security in the software-related industry. Early identification methods require significant manual efforts in crafting features or annotating vulnerable code. Although the recent pre-trained models alleviate this issue, they overlook the multiple rich structural information contained in the code itself. In this paper, we propose a novel Multi-View Pre-Trained Model (MV-PTM) that encodes both sequential and multi-type structural information of the source code and uses contrastive learning to enhance code representations. The experiments conducted on two public datasets demonstrate the superiority of MV-PTM. In particular, MV-PTM improves GraphCodeBERT by 3.36% on average in terms of F1 score.

Keywords: Pre-trained model · Vulnerability identification · Contrastive learning

1 Introduction

Code vulnerabilities are a major threat to the software-related industry. It is reported that the number of vulnerabilities has grown from 4,600 in 2010 to 175,477 by 2022¹. The number of vulnerabilities is still rapidly increasing.

Accordingly, the field of vulnerability identification is under intensive exploration in academia. In early-stage research, vulnerability identification methods can be categorized into three types: static analysis [1, 22], dynamic analysis [10], and machine learning methods [5, 18] based on hand-crafted features. Yet, a drawback of these methods restrains their performance. That is, they require vulnerability-related expertise and significant manual efforts, yielding them hard to be deployed and poorly scalable [24].

In later-stage research, researchers applied deep learning to address the aforementioned drawback existing in early-stage research [13]. Some studies [12, 24] leveraged several state-of-the-art deep learning techniques, e.g., LSTM and

¹ <http://cve.mitre.org/>

GGRN. A common feature of these methods is that they require large amounts of labeled data to perform supervised training and achieve better performance than conventional methods. Unfortunately, there is currently a lack of data annotated with vulnerability categories, and manually annotating data is labor-intensive. This hinders the further development of these methods in vulnerability identification.

The emergence of pre-training techniques alleviates the aforementioned problem. Thanks to the advancement, some pre-trained models for source code have been proposed, such as CodeBERT [3] and CodeT5 [21]. However, a significant disadvantage of these methods is that they ignore rich structural information such as abstract syntax and control flow. As such, a natural research question arises: how to combine multiple structural information with pre-trained models for vulnerability identification.

To tackle this question, we propose a novel Multi-View Pre-Trained Model (MV-PTM). Based on the pre-trained model, MV-PTM encodes both sequential information and multi-type structural information of the source code in a unified framework. Specifically, it generates representations of code under different structural information constraints. We term these representations as multiple views of source code. In this work, we use analysis tools to extract the Abstract Syntax Tree (AST), Control Flow Graph (CFG), and Data Flow Graph (DFG) of the source code and represent them as adjacency matrices that are taken as input by Structural-Aware Self-Attention Encoder to produce views containing different semantics. MV-PTM makes vulnerability predictions based on these views. In addition, MV-PTM uses contrastive learning [15] method for representation enhancement of structural information.

Our contributions are listed as follows:

- We propose a novel approach based on the pre-trained model that learns different structural information of the source code in a unified framework, which endows our model with the capability to represent the semantics of code more accurately.
- We perform contrastive learning based on multiple views of code to improve the performance of code representation learning, which is demonstrated to be better at characterizing code in the experiment.
- MV-PTM outperforms the state of the arts significantly with an average of 3.85% higher Accuracy and 6.80% F-1 Score.

2 Methodology

Overview. Figure 1 shows the overview of MV-PTM. First, we use Tree-sitter² to parse the source codes and get the code structural graphs. We then convert these graphs into adjacency matrices to guide the generation of multi-view code representations based on the Structural-Aware Self-Attention Encoder and pre-trained model. Afterward, the multi-view representations are fed to a Pooling Layer and Multi-layer Perceptron (MLP) for identification.

² <https://github.com/tree-sitter/tree-sitter-c>

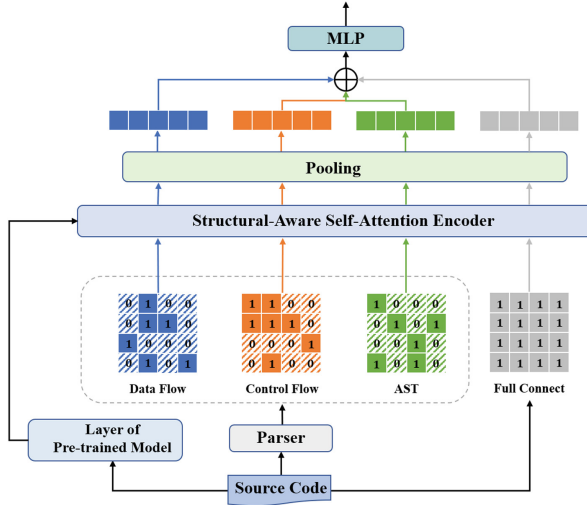


Fig. 1. The architecture of MV-PTM. In the adjacency matrix, 1 denotes there is an edge between corresponding nodes of code, and 0 otherwise. The layer of pre-trained model is used to obtain the source code embedding.

2.1 Structural Information

As aforementioned, we first obtain different structural graphs: CFG, AST, and DFG. Each node in these graphs represents a program statement, and each edge represents certain structural information. It should be noted that a pair of nodes may be connected by multiple edges because they have multiple dependencies. We use an adjacency matrix $M^{n \times n}$ to represent a certain type of edges in the graph (n is the number of tokens in the source codes). We set $M_{i,j} = 1$ if the i -th node and the j -th node are connected in the graph; Otherwise, $M_{i,j} = 0$.

CFG is a graphical representation of the paths that are traversed during the execution of a program. For example, as shown in Fig. 2(b), when the program executes the “if (a > 3)” statement, it decides whether “b = a - b” is executed according to the variable “a”.

AST is a tree-structured representation of the syntax structure of the source codes. Each node on the tree represents a syntactic structure. We use the subtrees in AST to analyze each statement in the program. Specifically, the tokens in the same statement can be connected to each other when constructing the adjacency matrix.

DFG tracks the use of variables during program execution, including access or modification of variables. Take Fig. 2(d) for instance, the variable “a” in “b = a - b” comes from “a > 3”.

2.2 Structure-Aware Self-attention Encoder

We utilize the pre-trained CodeBERT as the backbone in our approach to generate contextualized token representations, but our approach is flexible to other

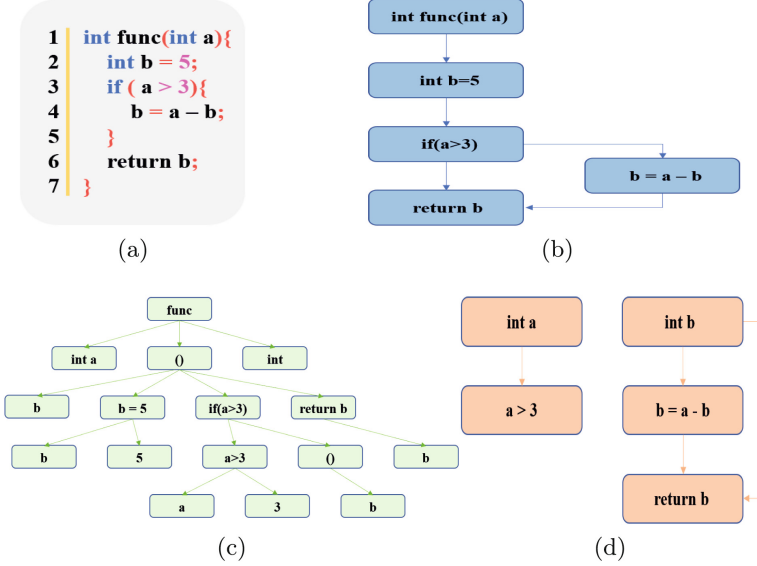


Fig. 2. An example of different structural adjacency matrices.

pre-trained models. Taking the source code x_i as an example, the representations Z_i are obtained by:

$$Z_i = \text{CodeBERT}(x_i). \quad (1)$$

On top of the backbone, we further design Structure-Aware Self-Attention Encoder (SASA) based on the self-attention mechanism proposed by [20]. SASA combines the structural information with the scaled dot-product attention using addition operations, given by:

$$\text{Attn}(Q, K, V, M) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}} + M\right)V, \quad (2)$$

where Q, K and V denote the query, key, and value matrix, respectively, and are set by Z_i . d_k is the dimension of K and softmax is Normalized Exponential Function. M is the adjacency matrix generated according to the specific structure information and it can constrain what the i -th token can attend to when computing attention values. Under the constraints of adjacency matrices, SASA can generate multiple views containing different structural information.

Under our observation, we notice that there are some similar dependencies between different structural information. Inspired by this, we make the node representation learning on different views share the same self-attention head. Besides, we also have the view-specific self-attention head. To fuse the representations learned from shared heads and view-specific heads, we use linear mapping to project them into the same space. As a whole, the SASA attention for one structural adjacency matrix (one view) is calculated as follows:

$$\text{SASA}(Q, K, V, M) = \text{Cat}(H_1, H_2)W^o, \quad (3)$$

where H_1 and H_2 correspond to the representation learned from the shared self-attention head and view-specific self-attention head, as shown in Eq. 2. Cat means the concatenation operation.

2.3 Multi-view Contrastive Learning

To enhance the representations learned from different structural information, we regard each type of information as one view and perform Contrastive Learning. This is motivated by the fact that different views of the same piece of code have some correlations and tend to cluster together in the semantic space.

To realize contrastive learning, we consider different views of the same code as positive pairs and those of different codes as negative pairs (Fig. 3). The loss function w.r.t. contrastive learning is expressed as:

$$\mathcal{L}_{CONTRA} = \psi_{ast} + \psi_{dfg} + \psi_{cfg}, \quad (4)$$

where ψ_{ast} takes AST as the Matched Structural view, and it is analogous to ψ_{dfg} and ψ_{cfg} . ψ is Normalized Temperature Scaled Cross Entropy Loss [2].

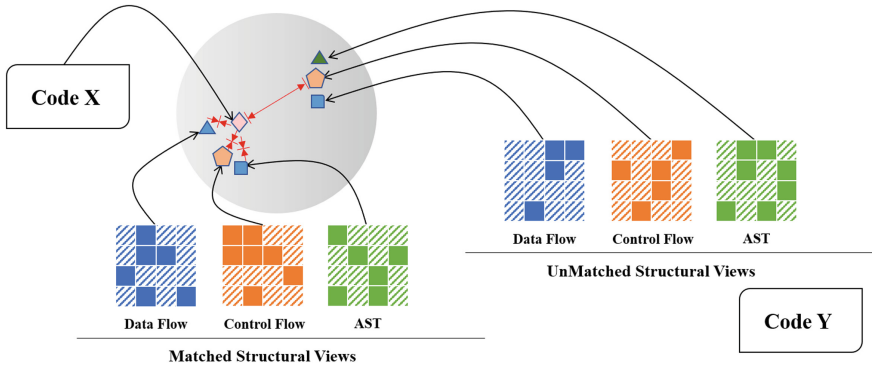


Fig. 3. Diamonds, triangles, pentagons, and squares correspond to code sequence, DFG, CFG, and AST, respectively. The circle denotes the semantic space.

2.4 Training Loss

We leverage Cross Entropy for training the main task, i.e., vulnerability identification, and the total loss for fine-tuning MV-PTM is given by:

$$\mathcal{L} = \mathcal{L}_{CLS} + \lambda \mathcal{L}_{CONTRA}, \quad (5)$$

where λ is the hyper parameter and we set $\lambda = 1$ in the experiments.

3 Experiments

3.1 Experimental Setup

Datasets. We evaluate our approach on two C-language datasets used in previous studies [21, 24], which contain manually-labeled functions collected from open-source projects FFmpeg and QEMU (Table 1).

Since some code snippets in the dataset exceed the length limit of CodeBERT, we discarded the codes that have more than 512 tokens. For long code segments, the recognition accuracy of MV-PTM is not ideal.

Table 1. Statistics of the datasets.

	FFmpeg	QEMU
Training set	3958	10903
Validation set	462	1378
Test set	499	1319
Total	4919	13600
Average length	274.5	325.3

Baselines. We choose the following six methods as the baselines since they represent the most up-to-date vulnerability identification mechanisms:

VulDeePecker [12]: It turns the source codes into a token sequence. The initial embeddings of tokens are trained via Word2Vec [14].

CNN [16]: It models the source codes as natural language and applies CNN to extract features from the code. The embedding initialization is the same as that of VulDeePecker.

Devign [24]: It represents the source code with the code property graph (CPG) which integrates all syntax and dependency semantics. Based on the graph, it uses Gated Graph Recurrent Network [11] for graph-level classification.

SELFATT [20]: Similar to [12], it takes the source code as sequences and exploits the multi-head attention mechanism for code representation learning.

CodeBERT [3]: It is a pre-trained model for programming language which has achieved acceptable performance on many code-related tasks such as code search and code documentation generation.

GraphCodeBERT [6]: It is the first pre-trained model that leverages code structure to learn code representation to improve code understanding.

3.2 Experimental Results

Performance Comparison. As shown in Table 2, MV-PTM outperforms all baseline methods on both two datasets. According to the experimental results, we summarize the following findings:

The local and structural characteristics of the code can improve the performance of vulnerability identification. Comparing CNN with

Table 2. Experimental results on the two datasets.

Methods	FFmpeg		QEMU	
	Accuracy	F-1 Score	Accuracy	F-1 Score
VulDeePecker [12]	0.5622	0.5923	0.5956	0.5644
CNN [16]	0.6032	0.6278	0.6482	0.3974
Devign [24]	0.5904	0.6015	0.6039	0.3244
SELFATT [20]	0.6152	0.6323	0.6361	0.3701
CodeBERT [3]	0.6353	0.6431	0.6907	0.6102
GraphCodeBERT [6]	0.6613	0.6724	0.6975	0.6497
MV-PTM	0.6874	0.6843	0.7149	0.7049

VulDeePecker, we can find that the Accuracy is significantly improved in both two datasets, implying that the local characteristics learned by CNN are indeed helpful for vulnerability identification. GraphCodeBERT outperforms CodeBERT with an average of 1.64% higher Accuracy and 3.44% F-1 Score.

MV-PTM performs best among all methods. MV-PTM has further improved its performance based on CodeBERT. It is noteworthy that the F-1 Score of baselines on the QEMU dataset is not ideal, while MV-PTM raised the F-1 Score to 0.7049.

3.3 Ablation Study

In this section, we verify the effectiveness of the three structural information and contrastive learning methods used in our work according to the results of the ablation study. Table 3 shows the experimental results.

Table 3. Effect of structural information and contrastive learning.

Methods	FFmpeg		QEMU	
	Accuracy	F-1 Score	Accuracy	F-1 Score
MV-PTM	0.6874	0.6843	0.7149	0.7049
MV-PTM w/o CFG	0.6553	0.6643	0.6983	0.6865
MV-PTM w/o AST	0.6573	0.6674	0.7036	0.6845
MV-PTM w/o DFG	0.6513	0.6683	0.6990	0.6892
MV-PTM w/o Contrastive	0.6693	0.6845	0.7005	0.6688

Different structural information improves the performance of the model, but to varying degrees. It can be observed from Table 3 that when any kind of structural information is removed, the performances of the model on both datasets decrease significantly.

Contrastive Learning can learn a more accurate multi-view representation of source code. We find that after removing the contrastive learning module, the performance of the model also decreases to a certain extent. This phenomenon implies that the contrastive learning method we proposed can make the model learn different code structure information more effectively.

4 Related Work

Vulnerability Identification. In academia, there usually are rule-based and learning-based methods for vulnerability identification. Rule-Based methods are widely explored in academia. SUTURE [23] is a static analysis method, which is capable of identifying high-order vulnerabilities in OS kernels. Learning-Based methods are a novel research direction that attracts much attention. VulDeeP-ecker [12] is an LSTM-based model, which represents the source code as vectors.

Pre-trained Model for Programming Languages. CodeBERT proposed in [3] is a bimodal model for programming language and natural language trained by Masked Language Modeling and Replaced Token Detection. GraphCodeBERT [6] considers the inherent structure of code by Edge Prediction and Node Alignment to support tasks like code clone detection [8, 9, 17].

Compared to CodeBERT, MV-PTM improves the accuracy by an average of 3.81% at the cost of 15% additional parameters and 25% training time, which is within acceptable limits.

Contrastive Learning. Contrastive learning is usually conducted in an unsupervised manner by increasing the similarity between the representations of positive pairs and decreasing the similarity between the representations of negative pairs [7, 19]. Data augmentation is a commonly-used technique to construct positive pairs, including rotation, scaling, and cropping in computer vision [2] and dropout in NLP [4].

5 Conclusion

In this paper, we propose MV-PTM, a pre-trained based model which uses structural information including AST, DFG, and CFG, to obtain multiple views of the source code. Besides, we introduce an auxiliary task based on contrastive learning to improve the performance of code representation. The experiments on two datasets demonstrate that structural information and contrastive learning are effective for vulnerability identification. In the future, we plan to explore the application methods of MV-PTM on long code segments and introduce the knowledge graph to generate reasonable explanations for the identified vulnerabilities.

Acknowledgment. This study was supported in part by the National Key R&D Program of China (2019YFB2102600), the National Natural Science Foundation of China (62002067), and the Guangzhou Youth Talent Program of Science (QT20220101174).

References

1. Chandramohan, M., Xue, Y., et al.: BinGo: cross-architecture cross-OS binary search. In: SIGSOFT FSE, pp. 678–689 (2016)
2. Chen, T., Kornblith, S., Norouzi, M., et al.: A simple framework for contrastive learning of visual representations. In: ICML, pp. 1597–1607 (2020)
3. Feng, Z., Guo, D., Tang, D., et al.: CodeBERT: a pre-trained model for programming and natural languages. In: EMNLP, pp. 1536–1547 (2020)
4. Gao, T., Yao, X., Chen, D.: SimCSE: simple contrastive learning of sentence embeddings. In: EMNLP, pp. 6894–6910 (2021)
5. Grieco, G., Grinblat, G.L., Uzal, L.C., et al.: Toward large-scale vulnerability discovery using machine learning. In: CODASPY, pp. 85–96 (2016)
6. Guo, D., Ren, S., Lu, S., et al.: GraphCodeBERT: pre-training code representations with data flow. In: ICLR (2021)
7. He, K., Fan, H., Wu, Y., et al.: Momentum contrast for unsupervised visual representation learning. In: CVPR, pp. 9726–9735 (2020)
8. Huang, C., Zhou, H., Ye, C., et al.: Code clone detection based on event embedding and event dependency. CoRR (2021)
9. Ji, X., Liu, L., Zhu, J.: Code clone detection with hierarchical attentive graph embedding. *Int. J. Softw. Eng. Knowl. Eng.* **31**, 837–861 (2021)
10. Li, Y., Chen, B., Chandramohan, M., et al.: Steelix: program-state based binary fuzzing. In: ESEC/SIGSOFT FSE, pp. 627–637 (2017)
11. Li, Y., Tarlow, D., Brockschmidt, M., et al.: Gated graph sequence neural networks. In: ICLR (2016)
12. Li, Z., Zou, D., Xu, S., et al.: VulDeePecker: a deep learning-based system for vulnerability detection. In: NDSS (2018)
13. Lin, G., Wen, S., Han, Q., et al.: Software vulnerability detection using deep neural networks: a survey. *Proc. IEEE* **108**, 1825–1848 (2020)
14. Mikolov, T., Sutskever, I., Chen, K., et al.: Distributed representations of words and phrases and their compositionality. In: NeurIPS, pp. 3111–3119 (2013)
15. van den Oord, A., Li, Y., Vinyals, O.: Representation learning with contrastive predictive coding. CoRR (2018)
16. Russell, R.L., Kim, L.Y., Hamilton, L.H., et al.: Automated vulnerability detection in source code using deep representation learning. In: ICMLA, pp. 757–762 (2018)
17. Sheneamer, A., Roy, S., Kalita, J.: An effective semantic code clone detection framework using pairwise feature fusion. *IEEE Access* **9**, 84828–84844 (2021)
18. Shin, Y., Meneely, A., Williams, L.A., et al.: Evaluating complexity, code churn, and developer activity metrics as indicators of software vulnerabilities. *IEEE Trans. Software Eng.* **37**, 772–787 (2011)
19. Tang, X., Dong, C., Zhang, W.: Contrastive author-aware text clustering. *Pattern Recognit.* **130**, 108787 (2022)
20. Vaswani, A., Shazeer, N., Parmar, N., et al.: Attention is all you need. In: NeurIPS, pp. 5998–6008 (2017)
21. Wang, Y., Wang, W., Joty, S.R., et al.: CodeT5: identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. In: EMNLP, pp. 8696–8708 (2021)
22. Xu, Z., Kremenek, T., Zhang, J.: A memory model for static analysis of C programs. In: ISoLA, pp. 535–548 (2010)
23. Zhang, H., Chen, W., Hao, Y., et al.: Statically discovering high-order taint style vulnerabilities in OS kernels. In: CCS, pp. 811–824 (2021)
24. Zhou, Y., Liu, S., Siow, J.K., et al.: Devign: effective vulnerability identification by learning comprehensive program semantics via graph neural networks. In: NeurIPS, pp. 10197–10207 (2019)

Localization



Discover the ICS Landmarks Based on Multi-stage Clue Mining

Jie Liu^{1,2}, Jinfan Wang^{1,2}(✉), Peipei Liu^{1,2}, Hongsong Zhu^{1,2}, and Limin Sun^{1,2}

¹ School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

{liujie1,wangjinfan,liupeipei,zhuhongsong,sunlimin}@iie.ac.cn

² Institute of Information Engineering Chinese Academy of Sciences, Beijing, China

Abstract. In recent years, the rapidly increasing landscape of industrial control systems (ICS) devices has made the ICS geolocation more important. However, IP-based geolocation cannot provide high accuracy geographical locations for ICS devices. Commercial databases only provide coarse mappings between IP hosts and physical locations. Measured-based geolocation relies on the number of high-quality landmarks. In this paper, we present a novel framework called OSI-Geo for serving high-quality landmark mining of ICS devices. The main idea is that there are many location-indicating clues in the open-source information exposed by ICS devices, which can be utilized to find their physical locations. The OSI-Geo automatically collects location-indicating clues to generate ICS landmarks at large-scale. We conduct real-world experiments for validating the effectiveness and performance of our method. The results show that OSI-Geo can accurately collect clues with over 99% recall and precision. Based on those clues, 36,872 stable landmarks, covering 162 countries and 5,596 cities, are obtained. Among them, there are 30,290 (82%) fine-grained landmarks accurate to street-level at least. The accuracy of IP geolocation has been improved significantly based on the ICS landmarks. Thus, OSI-Geo achieves effectively landmark mining for ICS devices.

Keywords: ICS devices · IP geolocation · Landmark mining · Network measurement

1 Introduction

More and more industrial control systems (ICS), including supervisory control and data acquisition systems, distributed control systems, building automation systems and other control systems, have connected to the Industrial Internet [13, 14, 21]. The study of geolocation for online ICS devices is becoming an important topic. The lack of ICS landmarks will influence the geolocation accuracy of ICS devices directly, because the accuracy of IP geolocation based on the common

Supported by National Key Research and Development Program of China under Grant 2020YFB2103803.

inference methods seriously relies on the number and density of landmarks. Thus, the study of ICS landmark mining has a significant value for Industrial Internet geolocation.

The IP-based geolocation of ICS devices is to provide the mapping between IP address and the physical location in the form of the latitude and longitude, or the country, city and street. However, commercial geolocation databases can only provide coarse-grained location information. Previous works [5] show that these databases have various discrepancies at the city-level. Although some improved methods have been proposed in recent years [12, 17, 19], IP geolocation inference is heavily dependent on the number of high-quality landmarks. However, available landmarks are community-based, most of them are located on academic networks, leading to a limited scale in terms of their number and coverage. Thus, the existed landmarks would not support the ICS devices to obtain the high accuracy of geolocation.

The ICS devices connected to the Internet expose a lot of detailed open-source information which is highly related to their physical locations, including the names of streets, buildings, organizations, or other related descriptions. These information, collectively referred to location-indicating clues, can be collected from the banners of application layer protocols and the HTML webpages of built-in web servers, which are helpful to infer the physical locations of ICS devices. These ICS devices can be utilized as landmarks to improve the accuracy of IP-based geolocation. Based on this observation, we present a framework called OSI-Geo, which is able to collect the location-indicating clues from online ICS devices and utilize those clues to infer the physical locations automatically.

The rest of this paper is organized as follows. Section 2 provides the background and motivation. Section 3 describes the location-indicating clues miner component. Section 4 details the ICS landmarks generation based on these clues. Section 5 presents the experimental evaluation. Section 5.5 surveys the related work. Finally, Sect. 5.6 concludes our work.

2 Background and Motivation

ICS devices communicate over protocols specified by manufacturers, including Modbus, DNP3, BACnet, Siemens S7, Tridium Fox, et al. Besides, quite a few ICS devices have built-in web servers for remote management. These online ICS devices are visible, and their banners and webpages are accessible via IP hosts. Although the basic idea is intuitive, there are three major challenges in practice. First, we need to identify which ICS protocols and products contain location-indicating clues on their banners and webpages. Second, how to collect the banners and webpages from online ICS devices and extract the location-indicating clues. Lastly, most of the location-indicating clues are not explicit and detailed, how to utilize them to infer the physical locations of ICS devices.

2.1 Clues in Banner

After analyzing banners of each protocol, we found location-indicating clues in the banners of BACnet and Tridium Fox. By default, the BACnet is found on

UDP port 47808, and the Tridium Fox on TCP port 1911 or 4911. As Fig. 1 shows, the banners of ICS protocols, which are collected from Tridium Fox and BACnet devices, disclose the location and organizations respectively.


<pre>fox.version=s:1.0.1 id=i:38 hostName=s:192.168.1.149 hostAddress=s:192.168.1.149 app.name=s:Station app.version=s:3.8.111 vm.name=s:Java HotSpot(TM) Client VM vm.version=s:1.5.0_81-b06 os.name=s:QNX os.version=s:6.4.1 station.name=s:SIX_HUNDRED_ALDEN_ROAD lang=s:en</pre>	<pre>Instance ID: 20000 Object Name: Fairfield Inn Davenport Location: Fairfield Inn Davenport Vendor Name: Trane Application Software: v5.30.1409 (release)</pre>
	<pre>Instance ID: 10000 Object Name: Unity Surgical Center Vendor Name: Trane Application Software: v5.60.1563 (release) Firmware: 4.4.44_SCNbuild-TracerSC-DEVELOPMENT</pre>

Fig. 1. Examples of clues in banners of ICS protocols.

2.2 Clues in HTML

By analyzing the webpages from common ICS products, we found that there are five ICS products whose webpages expose location-indicating clues. Tridium Web, Webctrl and AsiControl are deployed in building management system. Acquisuite is designed to collect energy data from meters and environmental sensors. AceManager provides a graphical user interface for the configuration options of Sierra Wireless AirLink modems. As show in Fig. 2, the location description "Ross1552_OxonHillMD" and the geographical coordinate "32.00405, -102.18490" are disclosed on the webpages of Tridium Web and Ace-manager respectively.

Ross1552_OxonHillMD



Username:

Password:

Remember Me

(a) Webpage of Tridium Web

DEVICE STATUS	
Cell Info:	CellInfo: TCH: 1100 RSSI: -45 LAC: 27659 CellID: 202507274
LTE Signal Strength (RSRP):	-78
LTE Signal Quality (RSRQ):	-15
Location Fix:	Location Fix Acquired
Location (Lat, Long):	3200405, -10218490

(b) Webpage of Acemaneger

Fig. 2. Examples of Clues in built-in webpages of ICS devices.

2.3 ICS Geolocation Based on Clues

We propose OSI-Geo to automatically collect location-indicating clues from the banners and built-in webpages of online ICS devices and utilize them to infer the geographical locations. Figure 3 illustrates the architecture. There are two major components: the location-indicating clue miner (LCM) and the geographical location generation (GLG). The LCM collects the open-source information

exposed by online ICS devices and extracts location-indicating clues. The GLG utilizes clues to generate landmarks. The implementation of LCM and GLG are detailed in Sect. 3 and Sect. 4.

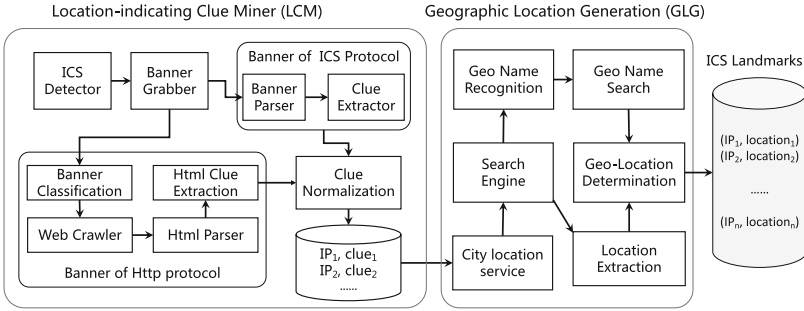


Fig. 3. Architecture of the OSI-Geo.

3 Location-Indicating Clue Miner

The LCM collects clues through 4 core modules: banner grabber, banner clue miner, HTML clue miner and clue normalization.

3.1 Banner Grabber

The banner grabber is designed to collect banners of online ICS devices and built-in web servers. We utilize ZMap [1] to scan ports 47080, 1911, 4911 and common HTTP ports across the entire public IPv4 addresses. For the host with one target port open, ZGrab is utilized to grab its banner [3].

3.2 Banner Clue Miner

The banners of BACnet and Tridium Fox have standard formats. We can use regex shown in Table 1 to extract clues on the parsed banners effectively.

Table 1. Regex for extracting banner clues.

ICS protocol	Regex
BACnet	(Object Name Location Description): (.*)\n
Tridium Fox	stationname=s:(.*)\n

3.3 Html Clue Miner

Since the banners of HTTP protocol do not provide the complete webpages, we need to scrape them by web crawler. A problem here is that the number of hosts with HTTP protocol is huge, while the banners of ICS built-in web servers only account for a small part. Hence, we need to screen HTTP banners related to the ICS products mentioned above.

Banner Classification. There are some features in the HTTP banners of ICS built-in web servers. (1) Generally ICS products use a special web server (e.g., "Niagara Web Server" for Tridium Web). (2) The webpages collected from the ICS devices in the same ICS product have a fixed HTML template. (3) The ICS built-in web servers commonly respond status code 301/302 and return a redirect URL for user login due to the authentication required.

Based on this observation, we consider this problem as a classification task and design a classifier shown in Fig. 4. We use BERT [2] as the classification model. The input text is merged by the response header and HTML title. The output will indicate whether this banner is related to one of those ICS products.

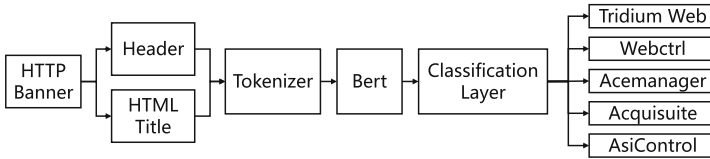


Fig. 4. The classification model for the banners of HTTP.

Clue Miner. The web crawler is utilized to collect the webpages screened by classifier. Another problem here is that the clues on the webpages are non-normalized and there are some unrelated organizations and facilities on the webpages, such as the manufacturers of ICS devices. Thus, utilizing Name entity recognition (NER) to extracted the clues has low recall and precision. Besides, due to the various product models of ICS devices and multiple versions of built-in web servers, generating regular expression rules manually to extract clue for all the webpages is a heavy workload.



Fig. 5. Two webpages using Niagara Web Server.

As mentioned above, the ICS webpages collected from the same ICS product have a fixed template and HTML structure. Figure 5 shows two webpages of Tridium Web deployed by the same Niagara web server, only the titles in the login panels are different. We leverage this observation and propose a diff-tags based clue extractor (DTCE) to generate rules for extracting clues automatically. The overview of DTCE is illustrated in Fig. 6, the DTCE first extracts the

tag structure of `<body>` in each webpage and filters out irrelevant tags such as ``, `<a>` and `<script>`. Then the webpages are clustered according to the same structure, and the tags with different contents in a same cluster are identified. Finally, the DTCE extracts the contents of those identified tags.

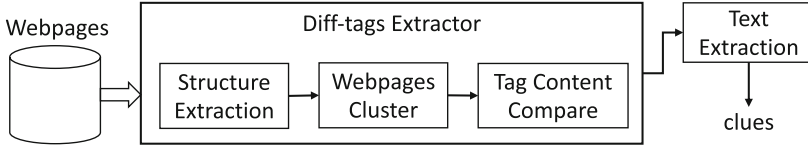


Fig. 6. Clue miner for built-in webpages of ICS devices.

3.4 Clue Normalization

There are some problems to determine the geographical locations of ICS devices using those clues directly: (1) Some clues are non-normalized. As shown in Fig. 5, the clues extracted from webpages of Tridium Web often use "CamelCase" or underscores as naming conventions to convey multiple-word names without using spaces. (2) The clues may contain some textual information which is irrelevant to the geographical locations, such as the manufactures, vendors, serial numbers of the ICS devices. (3) Since the hosts of ICS devices often open multiple ports for several services, there is more than one clue indicating to the same geographical location for the same host.

To address these problems, we first parse clues by splitting multiple-word names with space. Then we utilize our knowledge database to filter out the irrelevant text in clues. The knowledge database contains the manufactures, vendors, product descriptions and other information of the ICS devices. Finally, multiple clues of an ICS host are merged, and the duplicate clues are filtered out.

4 Geographical Location Generation

In this section, we present the design and implementation of GLG. The main idea is to leverage search engine and online map service to obtain the most relevant geographical locations for the clues.

Most clues only indicate the local locations without the cities where they are located. Thus, we first leverage commercial geolocation databases to determine the coarse-grained locations. To address the imprecise and inconsistencies problems of those databases, we leverage multiple databases and select the coincident location among them. For example, the results of "173.95.*.*" hosted by an Tridium Fox device are "Concord, North Carolina, US" in Maxmind GeoLite²¹, and "Charlotte, North Carolina, US" in IP2Location² and IPAPI³ we use "North Carolina, US" as the result.

¹ MaxMind GeoLite2. <https://www.maxmind.com/en/geoip2-databases>

² IP2Location. <https://www.ip2location.com>

³ IPAPI. <https://ipapi.com>.

Some of the clues contain detailed geographical location names, which can be utilized to query the geographical locations through online map service combined with the coarse-grained locations. However, the geographical names in quite a few clues are non-normalized, such as abbreviation and incomplete location names. Fortunately, the search engines, such as Bing and Google, can provide the relevant information for the query items. As an example shown in Fig. 7, the geographical location of clue "Providence UMC", collected from the Tridium Fox device which hosts "173.95.*.*", can be found in the Bing search results.

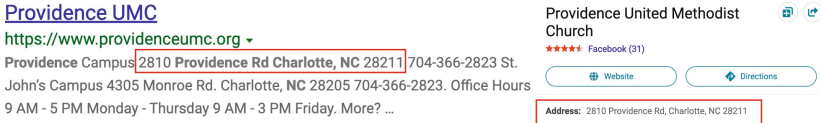


Fig. 7. Bing search results of "Providence UMC, North Carolina, US".

Based on this observation, we propose the search-engine based location generator (seLG), which is illustrated in Fig. 3. To find the geographical location of an given ICS device, the seLG first merges clue with coarse-grained location as the search query, and utilizes the search engine to scrape query results, which include a list of items and a possible map overlay. The title, geographical name and coordinates in the map overlay are extracted directly. Meanwhile, for each item, The geographical names in content are extracted by NER and converted to geographical coordinates by online map service. As a result, we obtain a list of geographical locations. Then the locations which are duplicate and conflicted with the coarse-grained location will be filtered out. If there is still more than one location, the term frequency-inverse document frequency (TF-IDF) [16] is utilized to measure text similarities of the clue and each title, and the most similar result is selected.

For providing normalized results and facilitating analysis, the GLG generates the ICS landmarks with the format [IP, geographical location name, lat/lon].

5 Experiment

In this section, we conducted real-world experiments. The OSI-Geo was utilized to collect the ICS landmarks from online ICS devices. We first constructed a dataset to validate the effectiveness of core modules in LCM. Then we analyzed the quality of the landmarks generated by GLG. Lastly, we leveraged a geolocation approach to evaluate the performance of geolocation with our landmarks.

5.1 Dataset

The LCM searched public IPv4 address for collecting banners of the specified application protocols, including BACnet, Tridium Fox and HTTP. We selected

3000 banners of HTTP. There are 2500 from the built-in web servers of ICS devices, and the rest are irrelevant. Each banner is labeled with its product. We scraped the webpages from the 2500 built-in web servers, and extracted clues manually. Then we utilized search engine and online map service to generate the geographical locations for each clue. The data for an ICS device was added to the dataset in the format [IP, banner, webpage, product, clue, geographical name, (lat/lon)].

5.2 Performance of LCM

HTML Banner Classification. The dataset is divided to training set (about 70%) and the validation set (the rest 30%). We fine-tune the BERT-based model using ADAMW optimizer with a batch size of 16 and the learning rate of $2e-5$. The dropout probability is kept at 0.1. We set the number of the epoch to 4 and save the best model. The max score is used to decide the product of a input HTML banner. If the max score is less than 0.9, we consider that this banner is irrelevant to any ICS product. Table 2(a) shows the performance on the validation set, our model achieved a precision of 99.54% and a recall of 99.66% on all the HTML banners of ICS product. The performance for identifying the banners of built-in web servers is perfectly acceptable in practice.

HTML Clue Extraction. We utilize the DTCE and a NER toolkit called Stanza [15] to extract the clues respectively. Noting that clues in the webpages of AceManager are geographical coordinates, we validate on the other four ICS products. As shown in Table 2(b), the NER cannot achieve both high recall and precision. For instance, the recall of NER is only 20.76% on the HTTP banners of Tridium Web, and the precision is only 37.95% on Acquisuite. By contrast, the performance of DTCE is very promising.

Table 2. Performance of HTML Clue Miner.

(a) Banner Classification				(b) Clue Extraction				
Product	Recall	Precise	F1-score	ICS Product	DTCE		NER	
					Recall	Precise	Recall	Precise
TridiumWeb	99.94%	98.97%	00.00%	TridiumWeb	99%	98.99%	20.76%	56.73%
Webctrl	93.67%	98.65%	00.00%	Webctrl	99.31%	99.42%	59.68%	85.71
Acquisuite	100%	100%	00.00%	Acquisuite	99.38%	99.84%	100%	37.95%
AsiControl	100%	99.19%	00.00%	AsiControl	100%	100%	47.66%	100%
Acemanager	100%	100%	00.00%	Total	99.24%	99.38%	54.2%	60.35%
Total	99.54%	99.66%	00.00%					

5.3 Landmarks Validation

Landscape. In total, We collected 36,052 banners from Tridium Fox, 12,641 banners from BACnet and 183,806,777 banners from HTTP. After classifying on these banners from HTTP, there are 132,794 banners from built-in web servers of ICS devices, whose webpages were scraped by web crawler. The LCM extracted 17,334 banners clues and 35,829 HTML clues. Table 3 lists the numbers of the clues extracted from each banner or webpage. After normalization, there are 37,691 IP-Clue pairs remaining. Based on those clues, the GLG generated 36,872 landmarks with unique IP addresses and their geographical locations. By analyzing the accuracy of geographical locations, we find that 30,290 landmarks (82%) are accurate to street-level. Among them, there are 26,872(73%) landmarks accurate to building level.

Table 3. The numbers of clues extracted from ICS banners and webpages.

Banner Clues		HTML Clues				
TridiumFox	BACnet	TridiumWeb	Webctrl	Acquisuite	AsiControl	Acemanager
12,937	4,397	29,287	3,808	493	107	2,134

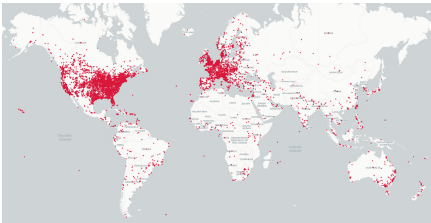


Fig. 8. Geographical distribution of the ICS landmarks.

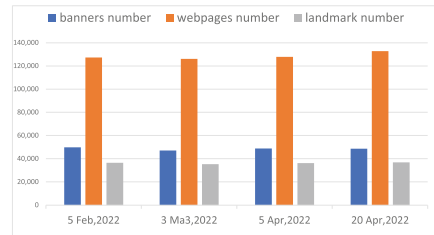


Fig. 9. Dynamic changes of the ICS banners and landmarks.

Geographical Coverage. We conduct an analysis on the geographical coverage of the ICS landmarks. Figure 8 depicts the world map, and the red dots represent the ICS landmarks. The landmarks cover a wide range of geographical locations, including 162 countries and 5,596 cities. Table 4 lists the top 10 countries and cities covered by ICS landmarks. About 80% landmarks are from North America and Europe.

Table 4. The Top 10 countries/cities of ICS landmarks.

(a) Top 10 countries				(b) Top 10 cities			
Country	Number	Country	Number	Country	Number	Country	Number
USA	22,906	UK	1,014	New York	1,995	Melbourne	316
Canada	2,127	Netherlands	823	Toronto	535	Chicago	291
France	1,390	Germany	303	Houston	527	London	246
Italy	1,188	Spain	281	Paris	475	Washington	232
Australia	1,048	Denmark	268	Sydney	397	Los Angeles	199

Stability. To validate the stability of ICS landmarks, we use OSI-Geo to collect banners and generate landmarks along with time. Figure 9 illustrates the dynamic changes using the data collected from February 5 to April 20. We can see that ICS landmarks remain a stable number. By comparing the landmarks collected in different time, we find that 75% of the landmarks are still available after 3 months, and 86% of them remain stable within two weeks, which demonstrates the long-term stability of the ICS landmarks.

5.4 Geolocation Performance

We experimentally validate IP geolocation performance using the ICS landmarks on two aspects. First, we implement a geolocation service for approximately pinpointing the geographical location of a given Internet host. Second, we compare the ICS landmarks with commercial geolocation databases.

IP Geolocation. We leverage a geolocation approach based on SLG [19] for the purpose of the geolocation experiments. The approach first sends the latency and topology probes to the target host and landmarks from our probe servers, then utilizes multilateration to shrink the region and pinpoint the target host to a landmark with the smallest relative latency. We collect 200 target hosts from M-Lab¹ and PingER² which have IP addresses and corresponding latitude/longitude pairs. Figure 10 shows the cumulative distribution function (CDF) of the geolocation errors on these target hosts. We observe that 92% of the hosts are less than 50KM error, of which 82.7% with less than 10KM error. The result indicates that the ICS landmarks can significantly improve the accuracy of geolocation.

¹ M-Lab. <https://www.measurementlab.net/>

² PingER. <https://www-iepm.slac.stanford.edu/pinger/>,

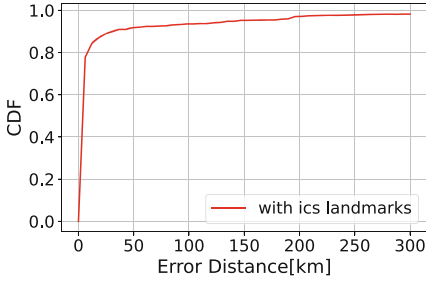


Fig. 10. IP geolocation performance

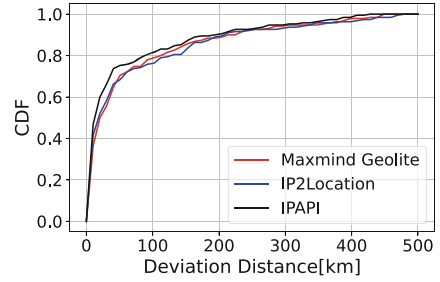


Fig. 11. Compared with geodatabases.

Geolocation Database. We select 200 target hosts with street-level geographical locations from the ICS landmarks. Each host has a detailed clue to make sure that its location is reliable. Figure 11 shows the CDF of deviation distances between the ICS landmarks and the locations in Maxmind GeoLite2, IP2Location and IPAPI. IPAPI achieves the highest performance, 47% of hosts have deviations less than 10KM and 75% less than 50KM.

5.5 Related Work

IP geolocation methods can be divided into data mining-based methods and measurement-based methods. Data mining-based methods aim to derive geographical location of an given IP address from public webpages or datasets. Measurement-based methods measure the latencies and topologies among VPs, landmarks and target hosts to estimate the geographical location of a given target host.

Data Mining-Based Methods. Huffaker et al. [8] leveraged geographical hints from hostnames to infer the geolocation of a large set of routers on the Internet. The accuracy is highly relied on the consistency of hints and actual locations. Liu et al. [11] leveraged check-in data from login logs of social networks to infer location of user’s IP address, this method requires specified logs and is not generic. Guo et al. [7] and Wang et al. [18] extracted the geographical information, including address information and the owner names of web servers, and derived the relationships with the IP addresses of web servers. However, due to the widely used cloud services and CDN, those landmarks have unstable and unverifiable issues. Wang et al. [19] proposed GeoCAM to monitor websites which host live webcams and extract the IP addresses and latitude/longitude of webcams for generating landmarks at large-scale. GeoCAM is able to generate landmarks with high accuracy and wide coverage.

Measurement-based Methods. Gueye et al. [6] proposed a constraints-based geolocation (CBG). The CBG converts the latencies between VPs and targets

to geographical distances and utilizes multilateration to locate the target host. Katz-Bassett et al. [9] leveraged network topology to improve the performance of CBG. Wong et al. [20] devised a general geolocation framework named Octant, which takes both positive and negative measurement constraints into account when estimating locations. Wang et al. [17] proposed SLG, which utilizes relative network distances and landmarks collected from webpages to achieve the street-level accuracy. Posit [4] and Soptter [10] utilized statistical analysis to estimate the relationship between network latencies and geographical distances.

5.6 Conclusion

In this paper, a novel IP landmark mining framework for the industrial control system (OSI-Geo) is proposed to discover the high accuracy landmarks of the online ICS devices. Firstly, an extraction method for protocol banners and built-in webpages of the ICS devices is put forward to gain a large number of ICS location-indicating clues. To avoid the impact of irrelevant named entity in the process of the clue extraction, we have designed the text-based rules generator. Then, a street-level geographical locations mining method is constructed by using the search engine and online map service. This method can infer the detailed street-level locations of ICS devices by some abbreviated and vague geographical clues. Based on the OSI-Geo, we have obtained 36,872 stable landmarks, covering 162 countries and 5,596 cities. There are 30,290 landmarks are fine-grained among them. Experimental results show that OSI-Geo can efficiently collect clues with over 99% recall and precision, and the landmarks can effectively improve the accuracy of IP geolocation. In the future work, we hope to improve geolocation accuracy for ICS devices by local landmark expand based on those ICS landmarks.

References

1. Adrian, D., Durumeric, Z., Singh, G.: Zippier ZMap: internet-wide scanning at 10 Gbps. In: 8th USENIX Workshop on Offensive Technologies (WOOT 14) (2014)
2. Devlin, J., Chang, M.W., Lee, K.: BERT: pre-training of deep bidirectional transformers for language understanding. arXiv preprint [arXiv:1810.04805](https://arxiv.org/abs/1810.04805) (2018)
3. Durumeric, Z., Adrian, D., Mirian, A.: A search engine backed by internet-wide scanning. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 542–553 (2015)
4. Eriksson, B., Barford, P.: Maggs: posit: a lightweight approach for IP geolocation. ACM SIGMETRICS Perform. Eval. Rev. **40**(2), 2–11 (2012)
5. Gharaibeh, M., Shah, A., Huffaker, B.: A look at router geolocation in public and commercial databases. In: Proceedings of the 2017 Internet Measurement Conference, pp. 463–469 (2017)
6. Gueye, B., Ziviani, A., Crovella, M.: Constraint-based geolocation of internet hosts. IEEE/ACM Trans. Netw. **14**(6), 1219–1232 (2006)
7. Guo, C., Liu, Y., Shen, W.: Mining the web and the internet for accurate IP address geolocations. In: IEEE INFOCOM 2009, pp. 2841–2845. IEEE (2009)

8. Huffaker, B., Fomenkov, M., Claffy, K.: Drop: DNS-based router positioning. *ACM SIGCOMM Comput. Commun. Rev.* **44**(3), 5–13 (2014)
9. Katz-Bassett, E., John, J.P., Krishnamurthy, A.: Towards IP geolocation using delay and topology measurements. In: *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pp. 71–84 (2006)
10. Laki, S., Mátray, P., Hága, P.: Spotter: a model based active geolocation service. In: *2011 Proceedings IEEE INFOCOM*, pp. 3173–3181. IEEE (2011)
11. Liu, H., Zhang, Y., Zhou, Y.: Mining checkins from location-sharing services for client-independent IP geolocation. In: *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, pp. 619–627. IEEE (2014)
12. Liu, J., Chang, W.C., Wu, Y.: Deep learning for extreme multi-label text classification. In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 115–124 (2017)
13. McLaughlin, S., Konstantinou, C., Wang, X.: The cybersecurity landscape in industrial control systems. *Proc. IEEE* **104**(5), 1039–1057 (2016)
14. Mirian, A., Ma, Z., Adrian, D.: An internet-wide view of ICS devices. In: *2016 14th Annual Conference on Privacy, Security and Trust (PST)*, pp. 96–103. IEEE (2016)
15. Qi, P., Zhang, Y., Zhang, Y., Bolton, J., Manning, C.D.: Stanza: a Python natural language processing toolkit for many human languages. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations (2020)*. <https://nlp.stanford.edu/pubs/qi2020stanza.pdf>
16. Tata, S., Patel, J.M.: Estimating the selectivity of TF-IDF based cosine similarity predicates. *ACM SIGMOD Rec.* **36**(2), 7–12 (2007)
17. Wang, Y., Burgener, D., Flores, M.: Towards street-level client-independent IP geolocation. In: *8th USENIX Symposium on Networked Systems Design and Implementation (NSDI 11)* (2011)
18. Wang, Y., Wang, X., Zhu, H., Zhao, H., Li, H., Sun, L.: ONE-Geo: client-independent IP geolocation based on owner name extraction. In: Biagioni, E.S., Zheng, Y., Cheng, S. (eds.) *WASA 2019. LNCS*, vol. 11604, pp. 346–357. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-23597-0_28
19. Wang, Z., Li, Q., Song, J.: Towards IP-based geolocation via fine-grained and stable webcam landmarks. In: *Proceedings of The Web Conference 2020*, pp. 1422–1432 (2020)
20. Wong, B., Stoyanov, I., Sirer, E.G.: Octant: a comprehensive framework for the geolocalization of internet hosts. In: *NSDI*. vol. 7, pp. 23–23 (2007)
21. Xu, W., Tao, Y., Guan, X.: The landscape of industrial control systems (ICS) devices on the internet. In: *2018 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (Cyber SA)*, pp. 1–8. IEEE (2018)

Mobility Models and Mobile Social Networking



Dynamic Mode-Switching-Based Worker Selection for Mobile Crowd Sensing

Wei Wang, Ning Chen, Songwei Zhang, Keqiu Li, and Tie Qiu^(✉)

College of Intelligence and Computing, Tianjin University, Tianjin, China
wwang@yeah.net, {chenning,zhangsongwei,qiutie}@ieee.org,
keqiu@tju.edu.cn

Abstract. Along with intelligent device popularization, mobile crowd sensing (MCS) has garnered considerable interest as a novel way of sensing data acquisition. Active and continuous worker engagement in tasks is a critical concern for sustainability when selecting workers to accomplish tasks in continuous MCS. Previous worker selection approaches are unsuitable for continuous MCS to ensure a large enough workforce. This paper proposes a framework for dynamic mode-switching-based worker selection called DMWS. DMWS lets temporary low-quality workers at tasks improve their competitiveness through hybrid mode switching based on task completion quality to ensure long-term sustainability. Therefore, they have the opportunity to be selected by the MCS platform again. The ultimate objective is to maximize space coverage at the lowest possible cost by increasing worker participation. As evidenced by experimental results on two real-world data sets, DMWS outperforms other methods in terms of space coverage under budget constraints.

Keywords: Mobile Crowd Sensing · Worker selection · Mode switching

1 Introduction

Mobile crowd sensing (MCS) is well suited for humongous sensing at the urban level [1]. Compared to the traditional sensing methods [2–4], MCS has garnered widespread interest because of its adaptability and low-cost benefits [5]. Pollution prevention [6] and traffic monitoring [7] are two common implementation situations. Completing these sensing activities necessitates the involvement of a wide range of workers. Workers do their tasks in two ways, depending on how much involvement they have: the opportunistic mode and the participatory mode [8]. Workers can perform tasks in the opportunistic mode by continuing their daily routines. While the cost is minimal, the region count covered by workers is limited. Workers walk to designated regions to execute tasks in the participatory mode. The fee is expensive because of transport expenses, but workers could cover more regions. If the two are mixed, it is referred to as the hybrid mode. It is vital to adapt the worker selection in hybrid mode to the dynamic MCS with a large number of workers' participation.

MCS is challenged with low worker engagement because of energy loss and labour time [9]. In the long term, reducing worker loss is critical to the system's sustainability [10]. Sustainability requires workers' active engagement in tasks for MCS to retain a sufficient number of workers on a limited budget. In specific worker selection methods, extra benefits are provided to workers who have already been unsuccessful in the bidding process to preserve them [10–12]. However, they pay little attention to low-quality workers. This paper uses quality to denote the percentage of workers' completed tasks. The workers' low-quality may be momentary due to inadequate device power, product defects, or unexpected worker interruption. It is unsustainable to drop or no longer select some workers because transient qualities. In particular, for activities such as environmental monitoring, it is essential to select suitable workers to make sure the data collected has high space coverage. Current studies of the hybrid mode aim to select workers who cover many locations without regard to sustainability [8, 13]. The MCS platform favours opportunistic workers who consistently deliver high-quality work. When no suitable opportunistic workers are available, expensive participatory workers are chosen. As a result, low-quality workers are not selected. They are unable to improve their qualities and eventually leave in frustration.

This study discusses the topic and presents DMWS, a dynamic mode-switching-based worker selection framework for MCS. DMWS intends to give chances for briefly low-quality workers to enhance their competitive advantage, thus expanding the engagement of prospective workers. As a consequence, task completion space coverage is improved and costs less. Because opportunistic workers have the cheap labour benefit, they are given greater priority than participatory workers for the same work. If the opportunistic worker picked was of poor quality, the task may not be finished. It is preferable to hire participatory workers. Therefore, while picking opportunistic workers, only high-quality workers are picked. Low-quality workers require the chance to enhance their competitiveness to become high-quality workers. Workers in DMWS can switch dynamically between two modes. Workers of poor qualities change to the participatory mode and have the option to enhance their qualities by performing low-cost tasks, reverting to the opportunistic mode.

2 Related Work

Sustainability is a critical research problem for continuous MCS. According to whether they alter their paths, workers' modes are classified as the opportunistic mode and the participatory mode. The opportunistic mode implies that workers accomplish tasks through their everyday routines [14, 15]. The participatory mode refers to workers who go to a specified region to execute assigned tasks. There are two types of approaches for assigning tasks to workers: worker-selected tasks (WST) [11, 16] and server-assigned tasks (SAT) [17, 18]. This paper discusses worker selection in WST. The participatory mode enables a large coverage region, whereas the opportunistic mode maximizes power and cost-efficiency [19]. As a result, combining the two modes makes sense. The ActiveCrowd framework was introduced by Guo et al. [20]. Time-sensitive tasks were assigned to

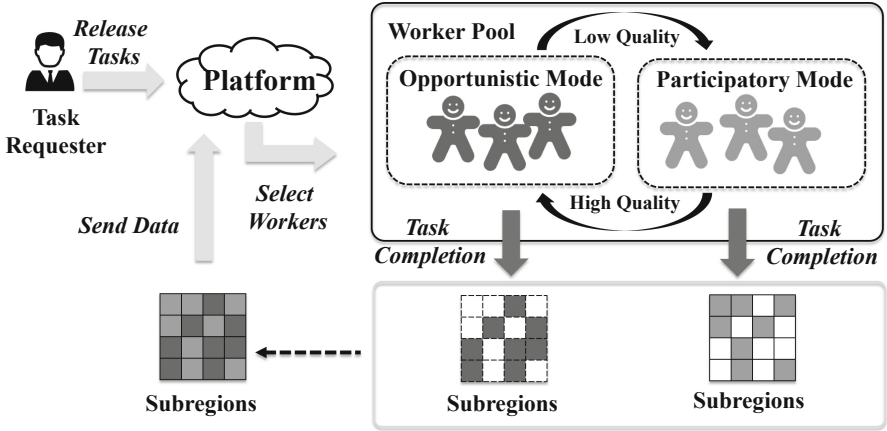


Fig. 1. The overview of DMWS.

participatory workers, while tasks that could wait were assigned to opportunistic workers. Wang et al. [8] conceived the Hytasker, the worker selection considering participatory workers while choosing opportunistic workers. In the first stage, tasks were assigned to opportunistic workers, whereas in the second stage, tasks were assigned to participatory workers. Previously proposed studies lack a method for adapting to continuous MCS to ensure sustainability in hybrid mode. When workers frequently do not get tasks, they will abandon the program in disappointment [9]. Current sustainability measures ensure MCS's long-term viability by offering additional benefits to unassigned workers [10, 11, 16, 21, 22]. However, they do not consider that when the desire for quality is vital, workers with poor qualities for transitory reasons would rarely be picked to execute tasks, resulting in workers quitting in despair. To solve the problem, this study introduces DMWS to maintain MCS sustainability.

3 System Model

As shown in Fig. 1, the MCS comprises three components: the task requester, the cloud platform, and workers. The task requester releases tasks to the platform. Workers carry out tasks in the opportunistic or participatory mode. Opportunistic workers record their predicted route on the platform before beginning each monitoring round. The platform identifies the target group of opportunistic workers. The other tasks are offered in reverse auctions to every worker. The platform selects workers based on the outcome of the reverse auctions. Additionally, workers can switch between the two completion modes based on their qualities. Finally, the data is analyzed and provided to the task requester.

The sensing cycle is divided into r rounds, expressed as $T = [t_1, t_2, \dots, t_r]$. The target subregion of the task is $A = [a_1, a_2, \dots, a_c]$. The worker set $W = [w_1, w_2, \dots, w_n]$ represents all workers in the worker pool. In each sensing round,

each subregion corresponds to a task. The subregion is considered covered when a worker completes a task in the subregion. In each round, the platform's goal is to cover as many subregions as possible. The set of workers selected to complete the task is $Y = [Y_1, Y_2, \dots, Y_r]$ of r rounds, which satisfies $Y \subseteq W$. $G_{i,j}(Y_i) = \{0, 1\}$ indicates the coverage of subregion j in the i -th round. When $G_{i,j}(Y_i) = 1$, subregion j is covered in round i . When $G_{i,j}(Y_i) = 0$, subregion j is not covered in round i .

In the i -th sensing round, if worker y is the opportunistic worker, the platform provides y with a reward $I_1(y, i)$; if worker y is the participatory worker, the platform provides y with a reward $I_2(y, i)$.

Based on the above, when the total budget for completing the task is *Budget*, the problem is defined as follows:

Maximize:

$$\frac{\sum_i^r \sum_j^c G_{i,j}(Y_i)}{r \times c} \quad (1)$$

Subject to:

$$\sum_i^r \sum_{y \in Y_i} I_1(y, i) + I_2(y, i) \leq \text{Budget} \quad (2)$$

4 Dynamic Mode-Switching-Based Worker Selection

As shown in Fig. 2, all opportunistic workers record their routes on the cloud platform before every sense round. It enables the platform to utilize Eq. (3) to assign every task to the most appropriate opportunistic workers who have the max S_1 of each region in the round r . Equation (4) indicates that the selected worker's incentive is $I_1(y_i, r)$. Workers who are not picked will get digital points in Eq. (5) for checking in at their pre-registration places, preventing them from quitting the MCS system. The number of digital points is initially fixed at 1.

$$S_1(w_i, r) = Q(w_i, r) + \lambda_1 \ln\left(\frac{d(w_i, r)}{m_i}\right) \quad (3)$$

$$I_1(y_i, r) = Re + \omega Q(y_i, r) \quad (4)$$

$$d_1(w_i, r) = e \quad (5)$$

where $Q(w_i, r)$ represents qualities of workers w_i , described below. λ_1 is the weight factor of the percentage of digital points $d(w_i, r)$ to the iteration the worker w_i has been chosen, m_i . Re is the base reward. The quality weighting parameter is ω . e is a positive constant.

The other tasks are provided to workers motivated to alter their future trajectories, including opportunistic and participatory workers. According to S_2 in Eq. (6), workers are chosen to accomplish tasks only when the bidding is within a certain amount, and the winners receive the payment specified in Eq. (7).

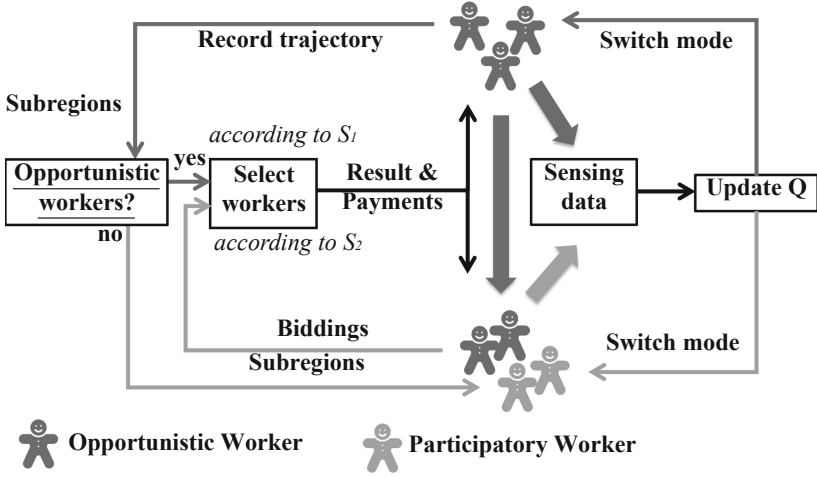


Fig. 2. Dynamic mode-switching-based worker selection process.

$$S_2(w_i, r) = \frac{Q(w_i, r)}{\text{bid}(w_i, r)} + \lambda_2 \ln\left(\frac{d(w_i, r)}{m_i}\right) \quad (6)$$

$$I_2(y_i, r) = \text{bid}(y_i, r) + \omega Q(y_i, r) \quad (7)$$

where $\text{bid}(w_i, r)$ is the worker w_i 's bid in the round r . λ_2 is the weight factor. The other parameters are the same as Eq. (3) and Eq. (4).

Workers who have not been picked will get the following digital points:

$$d_2(w_i, r) = e \cdot \frac{N_c(r)}{N_s(r)} \quad (8)$$

where e is a fixed number. $N_c(r)$ denotes the number of regions covered. $N_s(r)$ denotes the overall number of regions. Equation (8) increases the number of digital points available to workers bidding on assignments with few participants.

Mode switching is based on the quality of workers. The quality is determined by workers' performance, both short-term and long-term. The competence of a worker in the sensing round r is defined as:

$$z(w_i, r) = \frac{M_c(w_i, r)}{M_s(w_i, r)} \quad (9)$$

where $M_c(w_i, r)$ is the number of tasks completed by the worker w_i during the sensing round r . $M_s(w_i, r)$ denotes the aggregate number of the worker w_i 's assigned tasks during the sensing round r .

Short-Term Performance: The performance of the workforce may fluctuate over time. Recent time records have grown increasingly essential in reflecting worker capacities throughout time. Workers' short-term quality is stated as Eq. (10), using the index's most recent average value to give recent data greater weight.

$$q_s(w_i, r) = (1 - \alpha)^r + \sum_{t=1}^r \alpha(1 - \alpha)^{r-t} z(w_i, t) \quad (10)$$

where $(1 - \alpha)^r + \sum_{t=1}^r \alpha(1 - \alpha)^{r-t} = 1$. $0 < \alpha \leq 1$, and α is a factor relating to the rate of weight decay. r denotes the r -th sensing round currently in progress. The variable t is used to denote all prior sensing rounds.

Long-Term Performance: The workers' long-term performance is described as a combinatorial multi-armed bandit problem. Each worker serves as the arm of selection. The worker is compensated based on his or her performance during each round. The UCB method is used to represent the long-term performance of workers as Eq. (11). The algorithm uses the upper bound of the confidence interval to predict the quality of workers.

$$q_l(w_i, r) = \hat{z}(w_i, r) + \sqrt{\frac{\delta \cdot \ln(r)}{r_i}} \quad (11)$$

where $\hat{z}(w_i, r)$ is the average of all $z(w_i, r)$. δ is a positive constant. r_i represents the number of times the worker w_i has been chosen. r indicates that this is the r -th sensing round.

Using the concepts above, we define the quality as the following:

$$Q(w_i, r) = q_l(w_i, r) + \mu q_s(w_i, r) \quad (12)$$

where μ is a positive factor.

When qualities exceed or equal the threshold, workers switch to the opportunistic mode. When qualities are less than the threshold, workers switch to the participatory mode. By performing tasks in the participatory mode at a low cost, workers increase their qualities to revert to the opportunistic mode.

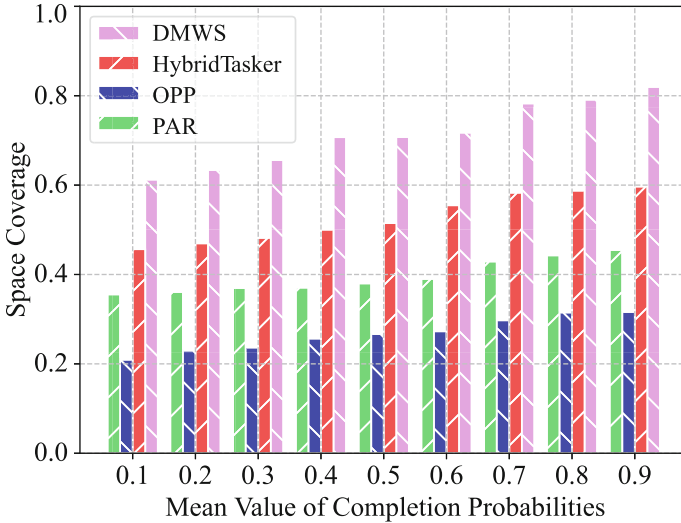
5 Performance Evaluation

This paper uses two publicly available location-based social network datasets: Foursquare [23] and Gowalla [24]. The two datasets provide information on users' check-ins and their friendship networks. Foursquare has 375 subregions, 577001 check-ins, 20613 friendship edges and 11173 users. Gowalla has 426 subregions, 311759 check-in records, 51774 friendship edges and 10868 users.

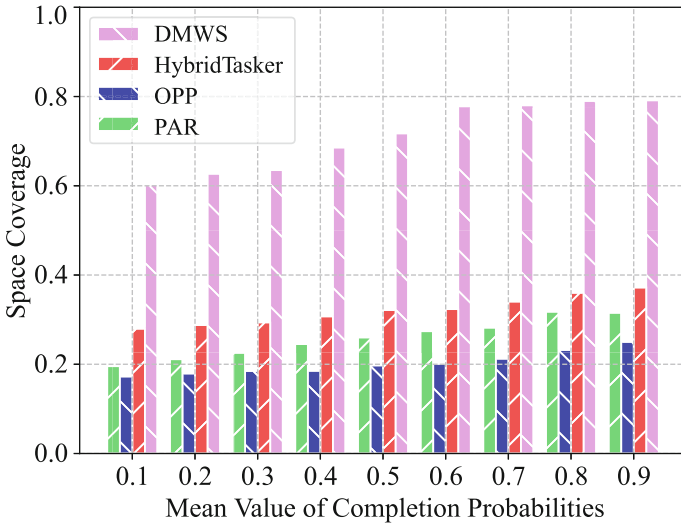
This paper uses three algorithms for comparison. HybridTasker adopts the same selection process as DMWS, but opportunistic workers with low qualities are obsoleted. OPP [25] employs a greedy approach to pick opportunistic workers within budget constraints. PAR [11] creates a winner selection mechanism to allocate sensing tasks to participatory workers.

The experiment's sensing tasks are a quarterly measuring plan. A quarter is considered 12 weeks. One day is a round for sensing. When workers are chosen, their completion probability is obtained using a Gaussian distribution. After

the mode switching, workers have a certain probability to increase Q to return to the opportunistic mode which is the switching probability. After switching, the completion probability increases by 50%. Each worker receives fewer than 5 tasks. The workers' bid range is [20, 30]. Other values are defined as 2 for Re , 0.1 for ω , 0.1 for λ_1 , 0.01 for λ_2 , 0.125 for δ , 0.5 for α , 1 for e , and 0.5 for μ .

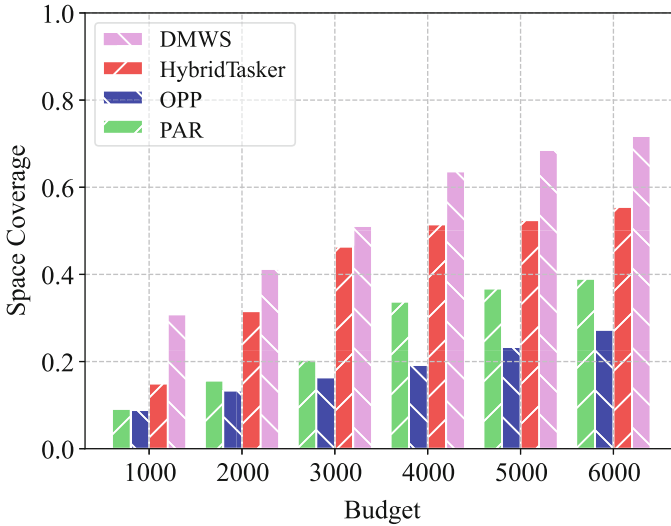


(a) Foursquare

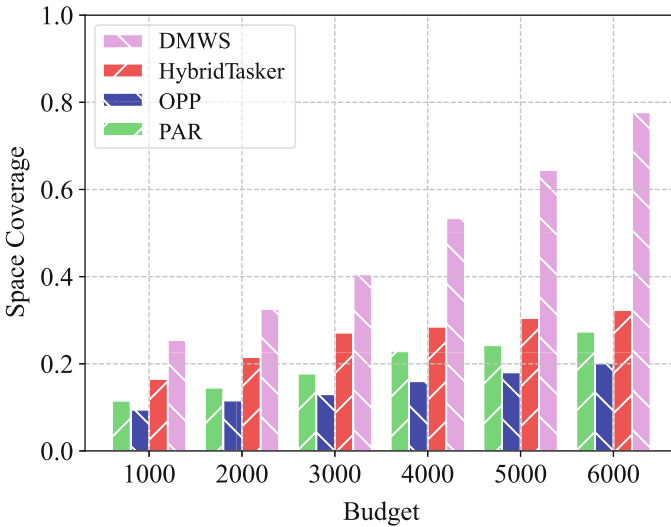


(b) Gowalla

Fig. 3. Space coverage comparison under various mean values of completion probabilities.



(a) Foursquare



(b) Gowalla

Fig. 4. Space coverage comparison under various budgets.

The first experiment is space coverage comparison under various mean values of completion probabilities. This paper sets the *Budget* to 6000 and the mode switching probability to 0.5. As seen in Fig. 3, DMWS outperforms other algorithms. The second experiment is space coverage comparison under various budgets. This paper sets the mean value of the completion probability to 0.6 and

the mode switching probability to 0.5. As seen in Fig. 4, DMWS achieves higher space coverage under the same completion budget than other algorithms.

6 Conclusion

In continuous MCS, temporary low-quality workers are difficult to be selected to complete tasks. To solve this problem, this paper proposes DMWS, a framework for dynamic mode-switching-based worker selection. In DMWS, temporary low-quality workers might improve their qualities by switching modes between the opportunistic mode and the participatory mode in order to earn the MCS platform's favour. Extensive simulation experiments on two real-world datasets demonstrate DMWS's good performance. The worker selection is part of the worker recruitment. In the future work, we will further explore the application of mode switching in worker recruitment.

Acknowledgement. This work is supported by the Joint Funds of the National Natural Science Foundation of China (No. U2001204), Tianjin Science Foundation for Distinguished Young Scholars (No. 20JCJQJC00250) and Key R&D Program of Tianjin (No. 20YFZCGX01150).

References

1. Li, X., Zhang, X.: Multi-task allocation under time constraints in mobile crowd-sensing. *IEEE Trans. Mob. Comput.* **20**(4), 1494–1510 (2019)
2. Qiu, T., Liu, J., Si, W., Wu, D.O.: Robustness optimization scheme with multi-population co-evolution for scale-free wireless sensor networks. *IEEE/ACM Trans. Networking* **27**(3), 1028–1042 (2019)
3. Chen, N., Qiu, T., Lu, Z., Wu, D.O.: An adaptive robustness evolution algorithm with self-competition and its 3d deployment for internet of things. *IEEE/ACM Trans. Networking* **30**, 368–381 (2021)
4. Qiu, T., Qiao, R., Wu, D.O.: EABS: an event-aware backpressure scheduling scheme for emergency internet of things. *IEEE Trans. Mob. Comput.* **17**(1), 72–84 (2017)
5. Gao, H., Feng, J., Xiao, Y., Zhang, B., Wang, W.: A UAV-assisted multi-task allocation method for mobile crowd sensing. *IEEE Trans. Mobile Comput.* (2022)
6. Huang, P., Zhang, X., Guo, L., Li, M.: Incentivizing crowdsensing-based noise monitoring with differentially-private locations. *IEEE Trans. Mob. Comput.* **20**(2), 519–532 (2019)
7. Farkas, K., Feher, G., Benczur, A., Sidlo, C.: Crowdsensing based public transport information service in smart cities. *IEEE Commun. Mag.* **53**(8), 158–165 (2015)
8. Wang, J., et al.: HyTasker: hybrid task allocation in mobile crowd sensing. *IEEE Trans. Mob. Comput.* **19**(3), 598–611 (2019)
9. Luo, T., Kanhere, S.S., Huang, J., Das, S.K., Wu, F.: Sustainable incentives for mobile crowdsensing: auctions, lotteries, and trust and reputation systems. *IEEE Commun. Mag.* **55**(3), 68–74 (2017)
10. Ji, G., Yao, Z., Zhang, B., Li, C.: A reverse auction-based incentive mechanism for mobile crowdsensing. *IEEE Internet Things J.* **7**(9), 8238–8248 (2020)

11. Yu, H., Yang, Y., Zhang, H., Liu, R., Ren, Y.: Reputation-based reverse combination auction incentive method to encourage vehicles to participate in the VCS system. *IEEE Trans. Netw. Sci. Eng.* **8**(3), 2469–2481 (2021)
12. Wang, H., Guo, S., Cao, J., Guo, M.: Melody: a long-term dynamic quality-aware incentive mechanism for crowdsourcing. *IEEE Trans. Parallel Distrib. Syst.* **29**(4), 901–914 (2017)
13. Lu, A., Zhu, J.: Worker recruitment with cost and time constraints in mobile crowd sensing. *Futur. Gener. Comput. Syst.* **112**, 819–831 (2020)
14. Xiong, H., Zhang, D., Chen, G., Wang, L., Gauthier, V., Barnes, L.E.: iCrowd: near-optimal task allocation for piggyback crowdsensing. *IEEE Trans. Mob. Comput.* **15**(8), 2010–2022 (2015)
15. Pu, L., Chen, X., Xu, J., Fu, X.: Crowd foraging: a QoS-oriented self-organized mobile crowdsourcing framework over opportunistic networks. *IEEE J. Sel. Areas Commun.* **35**(4), 848–862 (2017)
16. Gao, L., Hou, F., Huang, J.: Providing long-term participation incentive in participatory sensing. In: 2015 IEEE Conference on Computer Communications (INFOCOM), pp. 2803–2811. IEEE (2015)
17. Cheng, P., Lian, X., Chen, L., Shahabi, C.: Prediction-based task assignment in spatial crowdsourcing. In: 2017 IEEE 33rd International Conference on Data Engineering (ICDE), pp. 997–1008. IEEE (2017)
18. He, S., Shin, D., Zhang, J., Chen, J.: Near-optimal allocation algorithms for location-dependent tasks in crowdsensing. *IEEE Trans. Veh. Technol.* **66**(4), 3392–3405 (2016)
19. Tao, X., Song, W.: Location-dependent task allocation for mobile crowdsensing with clustering effect. *IEEE Internet Things J.* **6**(1), 1029–1045 (2018)
20. Guo, B., Liu, Y., Wu, W., Yu, Z., Han, Q.: ActiveCrowd: a framework for optimized multitask allocation in mobile crowdsensing systems. *IEEE Trans. Hum.-Mach. Syst.* **47**(3), 392–403 (2016)
21. Lee, J., Hoh, B.: Sell your experiences: a market mechanism based incentive for participatory sensing. In: 2010 IEEE International Conference on Pervasive Computing and Communications (PerCom), pp. 60–68. IEEE (2010)
22. Luo, T., Kanhere, S.S., Tan, H., Wu, F., Wu, H.: Crowdsourcing with tullock contests: a new perspective. In: 2015 IEEE Conference on Computer Communications (INFOCOM), pp. 2515–2523. IEEE (2015)
23. Yang, D., Qu, B., Yang, J., Cudré-Mauroux, P.: LBSN2Vec++: heterogeneous hypergraph embedding for location-based social networks. *IEEE Trans. Knowl. Data Eng.* (2020)
24. Cho, E., Myers, S.A., Leskovec, J.: Friendship and mobility: user movement in location-based social networks. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1082–1090 (2011)
25. Xiong, H., Zhang, D., Guo, Z., Chen, G., Barnes, L.E.: Near-optimal incentive allocation for piggyback crowdsensing. *IEEE Commun. Mag.* **55**(6), 120–125 (2017)



A Distributed Simulator of Mobile Ad Hoc Networks

Xiaowei Shu, Hao Wang^(✉), Zhou Xu, Kaiwen Ning, and Guowei Wu

School of Software Technology, Dalian University of Technology, Dalian, China
haowang@dlut.edu.cn

Abstract. The simulation of a Mobile Ad Hoc Network (MANET), before deployment or during the system running, provides a priori design validation and insightful observation of the real system. But existing simulation tools mainly enable these by means of centralized instead of distributed deployment, which in some sense, cannot truly replicate the real system settings. In this paper, we present a D**IS**tributively deployable Simulation tool for MANet (DISMAN), to accurately simulate MANET in a fully-distributed fashion thus allowing the emulation to scale with the network nodes without sacrificing accuracy. DISMAN is a fully functional tool that can be integrated with Kubernetes, support link layer (e.g., bandwidth limitation, delay, packet loss) and the multi-path as well as multi-hop transmission simulations. DISMAN is based on a four-layer architecture design, where on the top is a graphical user interface (GUI) layer for presentation and interaction. We further evaluate DISMAN with micro- and macro-benchmarks and show that DISMAN is easy to use and can assist MANET design by high level qualitative and quantitative simulations.

Keywords: Wireless mobile ad hoc network · Distributed simulation · Bandwidth limitation · Multi-path

1 Introduction

In recent years, with the development of wireless techniques, MANET technology has evolved as the foundation of more and more wireless mobile application scenarios. It deals with mobility and connectivity among network entities, for example, in the aspects like self-forming, self-healing, and peer-to-peer communicating, can thus be quickly deployed in complex and harsh environments [17]. A typical application scenario is unmanned aerial vehicle (UAV) swarm [13].

Evaluating MANET is challenging and expensive. Simulation is usually a less expensive option to validate new designs a priori or insightfully observe the running MANET systems. But many existing simulation platforms mainly provides centralized instead of distributed deployment, thus runs the node communication

This work was supported in part by the National Key Research and Development Program (No. 2020YFB2009503) and the Fundamental Research Funds for the Central Universities (No. DUT19RC(3)067).

and network forming within a single host. This, in some sense, cannot truly replicate the real system settings. Because it is in fact quite challenging to pack components into a single centralized node, especially when they are specific softwares or integrated network functions. The requirements to distributively deploy them into peer nodes during simulation are reasonable and even fundamental, since it is really difficult to accurately simulate the heterogeneous and complex software or function behaviors of an individual node in single node. A possibly more efficient way is straightforward deploying them into each physical (or virtual, in the sense of virtual machine) node instead of the simulated ones. Moreover, as a MANET is inherently a distributed system, a single host cannot fully capture the distributed system behavior, particularly in terms of communication processes. On the other hand, it is less efficient to transfer a design to a real system from a centralized simulation than its distributed counterpart.

Simulating the above stated network node independence is quite challenging. First, the communication network among nodes in the simulation is usually built over the underlying LAN networks, thus resulting in the bandwidth over-supply than the real system. We need to limit the link bandwidth according to the actual situation in the real system to be simulated. For example, in LAN the bandwidth is 1 Gbps, while in the simulation we must limit the bandwidth to 4 Mbps according to the real system settings. Second, it is also inherent that the nodes in a LAN can communicate to each other, while the real situation is that the network-forming depends on the movement of the nodes, wireless reachability, and even supervised on-off node states. It is also notable that these only relate to the physical connectivity, while networks can be logically divided into sub-networks according to node roles and functionalities. Third, end-to-end data transportation needs to consider the behaviors of different network layers, and all-stack protocol development is complicated. How to precisely abstract communication protocols or leave which part as open and user-configurable interfaces are very challenging. Two key factors to cover are multi-hop and multi-path transportation. Fourth, for better presentation and usability of the simulation, a GUI is necessary, but the data consistency and interaction efficiency between GUI and underlying simulation network need subtle maintenance.

For the acute need of simulating network nodes' independence and meeting above stated challenges, we propose DISMAN [1], a Distributively deployable Simulation tool for MANet, which allows users to build simulations with specific requirements. DISMAN can limit the bandwidth by splitting data into smaller scale and buffer-and-sending them in calculated longer time periods. The idea behind is based on the discrete time simulation and utilizes transport layer socket to imitate link layer behavior. Along with this idea, it can also simulate multi-hop and multi-path transmissions. We adopt the store-and-forward strategy at each node to supply data forwarding and multi-hop transportation. For simulating multi-path, we set up multiple sockets to imitate multiple physical network port and leave the network and transport layer settings defined by the users through user interfaces. Besides, DISMAN supports dynamic position and movement simulation, network topology updates, and the display of network status in the GUI windows. Further, for quickly deploying distributed front and back ends, DISMAN

provides scripts for install automation and support container-wise deployment. Finally, DISMAN has the corresponding statistics of multiple layers: throughput, per packet/hop and end-to-end delays, packet loss, etc.

This paper is organized as follows. We survey related work in Sect. 2. The design and system architecture of DISMAN are described in Sect. 3, with implementation details presented in Sect. 4. In Sect. 5 we evaluate the tool by inspect the functions and performance. Finally, we discuss limitations, future work, and conclude this paper in Sect. 6.

2 Related Work

There are some network simulation platform or dedicated tools for wireless MANET simulation. However, to accomplish distributed simulation, most of them exhibit good usability, though, have diverse disadvantages to overcome.

Dummynet [12], EmuSocket [8], DockEmu [20] can be options for link and network layers. Dummynet is an excellent open source link simulator, which has been directly embedded in Linux and MacOS. EmuSocket can better support statically defined point-to-point connections. DockEmu supports multi-node topologies. But they cannot simulate highly dynamic networks and need further implementation to support diverse requirements.

Some simulation tools are developed in the early stage, and there is no update or new version released recently, like GlomoSim [21], SWANS [9], JANE [7], GTNetS [18]. GlomoSim introduces the concept of grid, with which a simple entity can simulate several nodes in the system. SWANS is a wireless network simulator built on the JiST platform. JANE consists of both a simulation environment and an execution platform. Its main focus is that it allows the simulation code to be migrated to the real devices with little modification. The design concept of GTNetS is to identify nodes in the network as physical computers.

NS2 [5], NS3 [6], OMNET++ [2], and OPNET [4] are excellent network simulators. NS2 and NS3 are centralized multifunctional instead of distributed tool [19]. The framework of managing compute resources, adding components /protocols, and modifying components, is complex for users who simply want to simulate a mobile ad hoc network. The simulation of expediting for OMNET++ [10] intended to support wireless and mobile simulations within OMNET++. However, the mobility extension of OMNET++ is incomplete and it gives poor analysis of performance measures. OPNET is a widely used commercial network simulation platform, but not open-source and expensive.

So, we consider that it is necessary to design a tool for wireless MANET simulation, avoiding its physical layer characteristics, abstracting the object in the data link layer, covering as many wireless technologies as possible, providing good interfaces for users, to reduce the workload of the design of the simulation tool. Compared to Dummynet, DISMAN supports not only bandwidth restrictions and multi-hop transmission, but also multi-path transmission simulation. Moreover, DISMAN supports the simulation of node movement and network dynamic topology, with GUI presentation possibility. Compared to others, DISMAN is relatively light, easy to deploy and cover wider range of features.

3 Design of DISMAN

In this section, we describe the architecture and workflow of DISMAN.

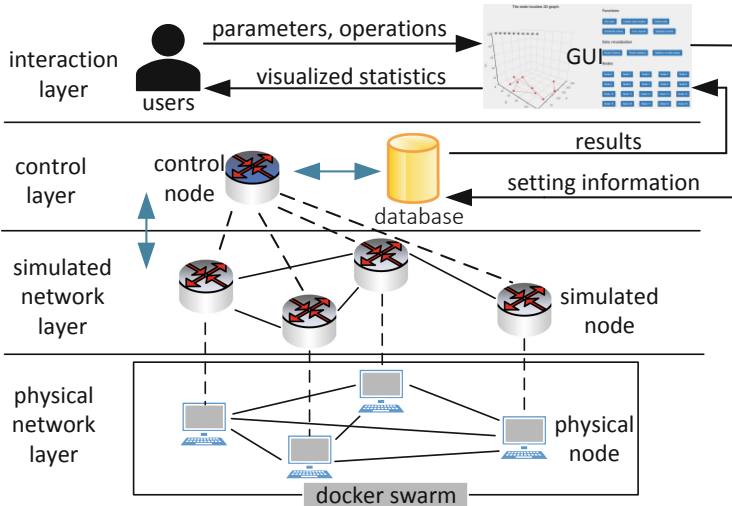


Fig. 1. A design of four-layer architecture.

3.1 Architecture

We depict four-layer architecture from bottom to top in Fig. 1, i.e., physical network layer, simulated network layer, control layer, interactive layer.

We build a distributed simulation physical architecture with containers (it can be transferred to physical machine settings almost with no cost), such that each node represents a moving individual in real system. These nodes have their own independent network characteristics, and also have some dynamic characteristics such as changing location, together constructing the physical network layer, as shown in Fig. 1. Note, we also introduce a “control node” to globally support the simulation process, which mainly supports routing, global control, data collection. It is regarded as “air”, i.e., serves only simulation, instead of representing some real physical nodes.

The simulation network layer is responsible for creating network nodes, containing network protocols, and simulating network dynamics like topology, network forming, and node movement. This is the main part of our design. Users can create multiple visual nodes in the simulation network layer for data interaction. Through this layer, users can limit the bandwidth and simulate the data transfer of wireless channel, sub-network division, topology reconstruction, multi-hop/multi-path transmission, delay measurement, etc. The realization of these functions will be introduced in details in next section.

The control layer is an “artificial” setting. As shown in Fig. 1, the function of the control layer is implemented based on the control node mentioned in the simulation physical layer. There is a database for data interaction, which maintains the adjacent matrix, data customization and collection of the simulation.

The interaction layer mainly implements three functions. First, users can set parameters used by the simulation, such as link bandwidth, node information, and the channel state. Second, users can through GUI set up experiments like multi-path and multi-hop transmissions. Third, users can visualize the collected statistics during and after the simulation results, export them, and have a real-time view on network topology dynamics and node movements.

3.2 Workflow

With the four-layer architecture mentioned above, the overall workflow of DISMAN is stated as below. See Fig. 1. Users can quickly deploy DISMAN with scripts. Users first browse the local address to enter the system observation window, then set the link bandwidth, number of initial nodes, locations, status, and other data on the GUI, afterwards wait for the network initialization to be completed. During the initialization of the simulation, the data set by the users in the GUI is mainly transmitted to the database in the control layer. The docker swarm [16] based on the servers respond and join the simulation network according to the information in the database.

The active nodes positions will be displayed in the three-dimensional coordinate system and refreshed dynamically in real time. Users can view the dynamic positions and the subnetwork forming. Users can click and view node information (location, status, etc.) and set node status, such as changing node information, adding nodes, deleting nodes, etc. Users can also control message sending, select source nodes, destination nodes and information, and manually set up data transmission, including multi-path and multi-hop transmissions. During the experiment, performance statistics of nodes are present in the GUI.

4 Implementation of Functional Modules

DISMAN can simulate many functions and entities of MANETs, such as wireless channel data communication, noisy channel, bandwidth setting, subnet division, network topology reconstruction, multi-hop/multi-path transmissions, and provide data collection and visualization.

4.1 Bandwidth Setting and Poor Channel Simulation

Bandwidth limitation is implemented using send/receive queues and send timing control, as shown in Fig. 2. When node 1 sends a message to node 2, DISMAN starts cutting the message according to the size of the data frame and delays each packet k by a time τ_k , where L_k is the length of the packet, b is the bandwidth of the link that users can set, t_p is the propagation delay of the link and t_q is queue

occupation when the packet was queued. Since the wireless link is unreliable, there would be packet loss, and the lost packets need to be retransmitted, but the retransmitted packets may still be lost. We set the upper limit of the number of packet retransmissions to N . If the packet is still lost after N retransmissions, then we can assume that the link is disconnected or over-congested. So τ_k can be expressed as follows $\tau_k = N \left(\frac{L_k}{b} + t_p \right) + t_q^{(N)}$. The noisy channel can reflect the reality of data transmission to some extent when simulating wireless channel data communication. To simulate the characteristics of noisy channels such as adjusting transmission delay and packet loss, DISMAN simulates random packet loss and enlargement of a set range based on the Bernoulli distribution, a discrete probability event before sending message. This can be parameterized.

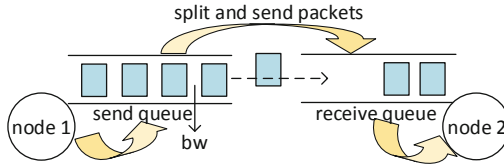


Fig. 2. The limitation of bandwidth.

4.2 Queue Management and Data Transmission

DISMAN adopts a default queue scheduling policy, i.e., FIFO [14], with size configurable either in bytes or number of slots. DISMAN simulates the data sending and receiving on the physical layer and the link layer with the socket function. Users can set network parameters, such as bandwidth, delay, queue size, source, destination nodes.

The data frame format in send and receive queues is composed of header, data and tail as shown in Fig. 3. Header includes flag and multi-hop information, where flag is data identity bit and multi-hop information contains the amounts of multi-hop nodes and ip/port of forwarding node. The data part is the metadata frame. The tail is made up of hostip, timestamp of sending (in μs), and node ID. Especially, node ID is used to simulate the MAC address of the network node.

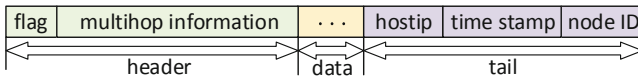


Fig. 3. The format of data frame.

The entire simulated system is obtained by processing ordered sequences of timestamped events, which are used to model packet transmission between communicating nodes. Each event is marked by a timestamp that specifies the simulation time at which the event must be processed, otherwise the simulation will not correctly simulate the system evolution. Therefore, in the process of simulating communication, DISMAN cuts and assembles the data packets according

to different separators, which include timestamps. With the timestamps of sending and receiving message, users can get the total delay of data transmission between two nodes, including sending delay, transmission delay, queuing delay and processing delay. After node i sends several messages to node j , the total delay T_i^j between node i and node j is $T_i^j = \sum_{k=0}^M \tau_k = t_j^l - t_i^f$, where t_j^l is the timestamp of the first frame in send queue of source node and t_i^f is the timestamp of last frame in receive queue of destination node, M is the number of data frames that the information is split.

4.3 Subnet Division

The communication between two nodes in the simulated network is almost certain to succeed, but in the real network, the transmission may fail simply due to a long distance, or node breakdown, etc., which will cause simulation errors. There are many subnet standards in the network. DISMAN sets the communication radius as the dividing standard in default and also the nodes have ON and OFF states. For other settings we can provide interfaces to accept as input. We define the node i state as “on” as $s_i = 1$ and “off” as $s_i = 0$. Therefore, if $s_i = s_j = 1$, the distance D_i^j between network nodes i and j is $D_i^j = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}$, otherwise $D_i^j = \infty$.

We define the default communication radius as R . And we use C_i^j as whether node j is within the direct communication range of node i , in other words, whether node i and node j can directly communicate instead of through other nodes. If $D_i^j \leq R$, $C_i^j = 1$, otherwise $C_i^j = 0$.

Since node cannot only communicate to other nodes directly, but also relay communication through intermediate nodes, the connectivity between nodes can be expressed as $C_i^j = I_{\{\sum_{n=0}^N (C_i^N \cdot \{\sum_{m=n}^N (C_n^m \cdot C_m^j)\})\}}$. Note that $I_{\{\cdot\}}$ is indicator function, where $I_{\{expr\}} = 1$ if ‘expr’ > 0 , and $I_{\{expr\}} = 0$ if ‘expr’ ≤ 0 .

DISMAN includes a pseudo-position update module. It uses Poisson probability distribution to add some randomness to the position initialization and control the track in a continuous way. Besides, it also provides API to support this kind of subnet related settings.

4.4 Multi-path and Multi-hop Networks

As mentioned in the subnet division, DISMAN has a pseudo-location update module that determines the reachability matrix according to the different location information of nodes, which gather on the control layer. DISMAN can support multi-hop networks with a shortest path algorithm based on the Dijkstra algorithm [11]. Each node will check the reachability matrix before sending data to decide whether to send. It also exposes program interfaces to ensure the extensibility of routing protocols.

We can see from the Fig. 4 that sending node, node 0, creates multiple socket interfaces with multithreading when node 0 sends data packets to multiple forwarding nodes, and then the data packets pass the forwarding node and arrive

at the destination node, node 4. So, DISMAN can support simulating multi-path transmission with socket function, simulating parallel with concurrency. There will be packet loss and data packet rearrangement in the transmission process, which is also in line with the actual network. Users can design data transmission protocols based on this to ensure that data packets arrive at the destination in the correct order finally. It also provides an interface for multi-path planning.

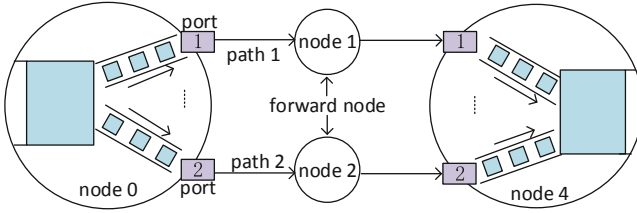


Fig. 4. The realization of multi-path transmission between node 0 and node 4.

4.5 Dynamic Network Reconstruction and Visual Interaction

Network topology reconstruction is a core section of MANET, considering ON and OFF state switch of nodes, high mobility of nodes, etc. In order to provide simulation of it, the location information of nodes is continuously collected and distributed on the control layer. Note, in dynamic reconfiguration, DISMAN does not refine the module of energy consumption. Users simulate the energy consumption of mobile nodes by controlling the state of nodes (ON and OFF).

The sending and receiving of simulation data is a distributed aggregation structure around the database. As mentioned above, the database belongs to the control layer. The GUI exposes a web-based interface to monitor and control the experiments, where users can intuitively view part of the simulation data by clicking the Buttons. DISMAN also provides visualization of network topology reconstruction with the 3D model of ECharts [3] and adding display windows.

5 Evaluation

In this section, we evaluate DISMAN. The results show that DISMAN can closely simulate the real system as it has a good support for distributed MANET in terms of node movement, link layers, network forming, and visualization. We start the evaluation with a series of micro-benchmarks that highlighting the individual features of DISMAN and justify the major design decisions in Sect. 5.1–Sect. 5.4. These experiments were carried out on a single server, a Dell PowerEdge R740 with 64 GB RAM. All virtual nodes run on Ubuntu Linux 20.04, kernel 5.4.0–100-generic with docker version 20.10.12. The database version is Mysql 5.8. Then, we compare DISMAN with other widely used tools.

5.1 Evaluation of Bandwidth Limitation

In the actual wireless network scenario, because of the environmental influence, the bandwidth cannot reach the theoretical value. The bandwidth mentioned here represents the average throughput that the user wants to set. We deploy 5 virtual network nodes with a default data frame size of 50 bytes. A transmission was established between the two virtual network nodes and different size messages were transferred and timed to compute the bandwidth. Take 1 MB, 5 MB, 10 MB message for 10 experiments based on permutations and record the corresponding average throughput as shown in Table 1, the average limit of throughput is around 15 Mbps. If the user wants to simulate a smaller bandwidth, to realize the limitation of small bandwidth, each frame is paused for a corresponding microsecond time before the transmission according to the bandwidth setting by the user and the processing delay of the code.

Table 1. Evaluation of bandwidth limitation.

Frame size	50 B			4 KB		
Message size	1 MB	5 MB	10 MB	1 MB	5 MB	10 MB
Bandwidth	15.2 Mbps	14.9 Mbps	15.3 Mbps	255.7 Mbps	256.5 Mbps	256.2 Mbps

If the user wants to simulate a larger bandwidth, the system supports modifying the data frame size. We set the frame size to 4 KB, and then repeat the above experiment. Statistics showed in Table 1 that the average throughput between two network nodes is around 256 Mbps. A bigger bandwidth is also possible.

5.2 Evaluation of Network Performance Parameters

The Statistics of Delay. Let 10 nodes join the network in turn. In order to test the visualization of delay conveniently, the information transmission between nodes is set as point-to-point form. The mobile node speed is 0 m/s, the location update frequency is once every 10 s, the node status is 1 (power on), and the 10 node locations are within the communication radius of 100 m, which ensures that the 10 nodes are always in the state of full connectivity. By default, the channel environment is noise-free, the entire simulated network bandwidth is 15 Mbps, and the data frame size is 50 bytes.

In order to test the average and the worst delay of node 0, we set the data flow direction as follows. All nodes except node 0 send 20 byte message to node 0 (which will be cut into two data frames). Check the delay of node 0 at the front end, as shown in Fig. 5. Repeat the above experiment again, but turn on the noise channel simulation function, set the maximum random increase of delay to 5ms and random packet loss. The node 0 delay status is as shown in Fig. 5. First, these two figures show the delay results to users. When simulating the noisy channel, the maximum and average delays from each node to node 0 increase significantly, node 2, 4, 5 and 7 have no data because of packet loss, and node 1, 8 and 9 lose some packets, so the maximum delay and average delay are equal. Figure 5 indicates the randomness and availability of the noisy channel.

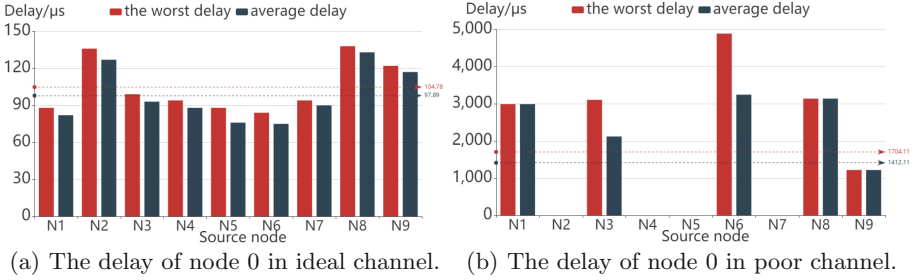


Fig. 5. The delay visualization of experiment.

The Statistics of Data Packets. The experiment is consistent with most of the parameters in the above experiment, and it is also divided into two simulation experiments of noisy channel and none noisy channel. To count more data packets, we set the information transmission path planning and data flow direction as follows. Except for node 0, all other nodes send 5 byte messages to node 0, such as “hello”. Before node 9 sends to node 0, the path request information is sent to the control plane by node 9, and the control plane sends the path planning information back to node 9. The path planning is that the multi-hop nodes in the path are 8, 7, 6, 5, 4, 3, 2, and 1 in turn, including the IP and port of the following multi-hop nodes. Then node 9 sends it to node 0 according to the routing information. The messages sent by other nodes to node 0 are similarly forwarded according to the planned path. For example, if node 5 sends messages to node 0, the multi hop nodes in the path are 4, 3, 2, and 1 in turn. After the test, view the data visualization page, and the data packets received by each node are shown in Fig. 6. It shows the packet counts to the users.

5.3 Evaluation of Multipath Transmission

We deploy four static virtual network nodes, node 0, node 1, node 2, node 3. Set some basic parameters, such as 15Mbps for the entire simulated network bandwidth and 50B for the frame size. First we set one data flow direction, node 0→node 1→node 3 and let node 0 transmit 10MB information to node 1 and forward from node 1 to node 2. Repeat the experiment 10 times and record

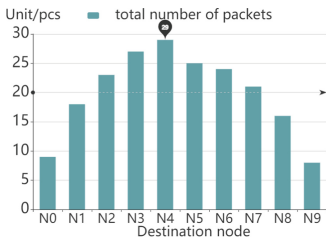


Fig. 6. Packet counts.

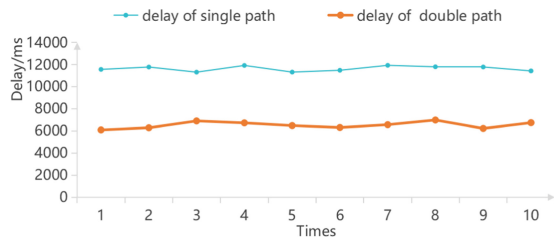


Fig. 7. The delay of single vs. double path.

the delay results. Then we set two data flow directions, node 0→node 1→node 3 and node 0→node 2→node 3. Then let node 0 concurrently transmit 5MB information to node 1 and node 2. To ensure a more accurate evaluation of multi-path transmission, we set the processing latency of forwarding nodes to 0.

As shown in Fig. 7, the delay of 10MB information in single-path transmission is concentrated in 11–12s, delay of 10MB information in double-path transmission is concentrated in 6–7s, and the delay of single-path transmission is approximately twice as long as that of double-path transmission. The experimental results show that the multi-path transmission module can effectively support simulation of multi-path transmission.

5.4 Evaluation of Dynamic Network

To evaluate the effective implementation of other functions, we design a network consisting of multiple mobile nodes. There are 20 peer-to-peer mobile nodes in the network. The default bandwidth of the simulation network is set to 15 Mbps, the data frame size is set to 50B, the communication radius is 100 m, each node moves randomly at a certain speed and the location update frequency is once in 3 s. The location updates information of all nodes converge to the control plane based on the pseudo location update module stated above. In our design, nodes 0–9 simulate moving objects on the ground, so the Y coordinate is set to 0, and the moving speed is 5 m/s, nodes 10–19 simulate flying objects in the air, and the speed is 20 m/s.

Our experimental steps are set as follows over time: At the beginning of the experiment, we successively add 10 nodes (nodes 0 to 9) to the network, and the location update is set to update every 3 s. All nodes join the network and start updating locations successively after 30s. For ease of description, we do not show the initial positions of the 10 nodes. At the 90s, we add 10 nodes (node 10 to node 19) to the network in turn. After another 90s, tool can record all location information, view the three-dimensional geographic display and the number of subnets on the visualization page, shown in Fig. 8.

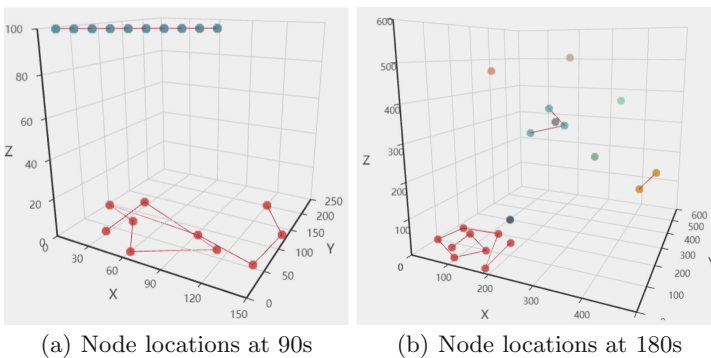


Fig. 8. The visual display of dynamic network in different time.

5.5 Performance of DISMAN

In this part, we mainly evaluate DISMAN from a macroscopic perspective. DISMAN is a distributed simulation environment, so it facilitates more realistic simulations than those widely used simulators such as NS2 and NS3, which are deployed in a single computer or server. Compared with dummynet, DISMAN provides more self-contained functions. It realizes the detailed simulations of link layer, such as bandwidth, delay, packet loss, queue management, etc., and further facilitates the simulation of multi-path concurrent transmission. Users do not need further implementations.

Besides, it is an easy-to-use tool. Its deployment and learning cost is very low. It supports the rapid deployment of mobile network simulation environment. Users can deploy the largest node of the required simulation network in a few minutes according to their needs. DISMAN also covers a visual GUI and supports users to interact directly on the GUI, which can not only support setting dynamic behaviors, such as modifying wireless network bandwidth, simulating noisy channel, and modifying the state of dynamic nodes, but also directly view the dynamic network topology in the 3D view. To compare, other tools like ns2, has more complicated installation and deployment. The deployment of mobile network simulation environment requires further implementation, either there is no visualization module that can be used directly.

In terms of scale, one sees nearly no limitation. DISMAN can deploy hundreds or thousands of network nodes on several servers to support large-scale deployment, using Docker. Mininet [15] also supports the simulation of hundreds of network nodes, but it is limited to single machine deployment. Moreover, the simulation of dynamic behaviors is badly supported.

6 Conclusion

In this paper we presented DISMAN, a distributively deployable simulation tool. It can better replicate system behavior of MANET by limiting the bandwidth, developing multi-hop and multi-path transmissions, providing dynamic node position and movement simulation, network topology updates, and the display of network status in the GUI windows. Moreover, DISMAN provides scripts for install automation and container-wise deployment. DISMAN provides an interactive GUI, where users can not only get statistics but also input parameters and control the simulation. Numerical experiments evaluated the functionalities of the tool and showed that DISMAN can become a powerful, easy to use, ready for exhibition, distributed simulation tool, efficiently assisting design of MANET.

References

1. <https://github.com/ADistributedNetSimulator/MANET.git>. Accessed 5 Aug 2022
2. <https://www.omnetpp.org>. Accessed 10 Apr 2022
3. <https://echarts.apache.org>. Accessed 10 Apr 2022

4. Home - opnet worldwide. <https://www.opnet-group.org>. Accessed 18 Mar 2022
5. The network simulator. ns-2. <https://www.isi.edu/nsnam/ns>. Accessed 15 Apr 2022
6. The network simulator. ns-3. <https://www.nsnam.org>. Accessed 15 Apr 2022
7. Newcastle university computing laboratory. Javasim's users guide. <https://archiveshub.jisc.ac.uk>. Accessed 20 May 2022
8. Avvenuti, M., Vecchio, A.: Application-level network emulation: the emusocket toolkit. *J. Netw. Comput. Appl.* **29**, 343–360 (2006)
9. Barr, R.: An efficient, unifying approach to simulation using virtual machines. Ph.D. thesis (2004)
10. Bautista, P.A.B., Urquiza-Aguiar, L.F., Cárdenas, L.L., Igartua, M.A.: Large-scale simulations manager tool for OMNeT++: expediting simulations and post-processing analysis. *IEEE Access* **8**, 159291–159306 (2020)
11. Candra, A., Budiman, M.A., Hartanto, K.: Dijkstra's and a-star in finding the shortest path: a tutorial. In: 2020 International Conference on Data Science, Artificial Intelligence, and Business Analytics (DATABIA) (2020)
12. Carbone, M., Rizzo, L.: Dummynet revisited. In: *ACM SIGCOMM Computer Communication Review (SIGCOMM)* (2010)
13. Fu, X., Zhang, J., Chen, J., Wang, S.: Formation flying and obstacle avoidance control of UAV cluster based on backbone network. In: *IEEE 16th International Conference on Control Automation (ICCA)* (2020)
14. Geyer, F., Scheffler, A., Bondorf, S.: Tightening network calculus delay bounds by predicting flow prolongations in the FIFO analysis. In: *IEEE 27th Real-Time and Embedded Technology and Applications Symposium (RTAS)* (2021)
15. Lantz, B., Heller, B., McKeown, N.: A network in a laptop: rapid prototyping for software-defined networks. In: *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*. Association for Computing Machinery (2010)
16. Merkel, D.: Docker: lightweight Linux containers for consistent development and deployment. *Linux J.* **239**(2), 2 (2014)
17. Rashid, M.M., Isawi, M., Mahmood, B.A.: An extensive analysis of the ad hoc network. In: *Proceedings of the 6th International Conference on Engineering & MIS(ICEMIS)*. Association for Computing Machinery, New York (2020)
18. Riley, G.F.: The Georgia tech network simulator. In: *Proceedings of the ACM SIGCOMM Workshop on Models, Methods and Tools for Reproducible Network Research*. Association for Computing Machinery (2003)
19. Saad, A., Osman, H., Ebedon, M.M.: Review of ad hoc networks scenarios and challenges in years 2015–2019. In: *International Journal of Electrical and Computer Engineering Systems(ICEMIS)*. Association for Computing Machinery (2021)
20. To, M.A., Cano, M., Biba, P.: Dockemu - a network emulation tool. In: *IEEE 29th International Conference on Advanced Information Networking and Applications Workshops*. Institute of Electrical and Electronics Engineers (2015)
21. Zeng, X., Bagrodia, R., Gerla, M.: Glomosim: a library for parallel simulation of large-scale wireless networks. In: *Proceedings. Twelfth Workshop on Parallel and Distributed Simulation PADS*. Institute of Electrical and Electronics Engineers (1998)



Social-Network-Assisted Task Selection for Online Workers in Spatial Crowdsourcing: A Multi-Agent Multi-Armed Bandit Approach

Qinghua Sima¹, Yu-E Sun²(✉), He Huang¹, Guoju Gao¹, and Yihuai Wang¹

¹ School of Computer Science and Technology, Soochow University, Suzhou, China

² School of Rail Transportation, Soochow University, Suzhou, China
sunye12@suda.edu.cn

Abstract. The popularity of smart devices and the availability of wireless networks bring considerable attention to Spatial Crowdsourcing (SC). Existing studies mainly focus on solutions to different optimization objectives of the SC platform, ignoring the entitlement of workers. This paper starts from the perspective of workers and investigates how to select suitable tasks for each online worker such that everyone can maximize their individual profit. Since the profit is related to the completion degree of tasks that is determined by the prior unknown parameter, we model the problem as a Multi-Agent Multi-Armed Bandit (MAMAB) problem. We propose a Payment-Estimation-Based Solution (PEBS), allowing workers to sequentially make decisions on task selection based on their observations and estimations. Specifically, the proposed PEBS first utilizes the social network among workers and assists workers in learning the information of tasks from the historical data. Then, it introduces the idea of probability matching in Thompson Sampling (TS) to help estimate the profit of workers and deal with the task selection problem. Finally, extensive simulations show that our proposed mechanism is efficient in optimizing the individual profit of workers.

Keywords: Social network · Multi-agent multi-armed bandit · Online task selection · Thompson sampling · Spatial crowdsourcing

1 Introduction

Spatial crowdsourcing (SC) [1, 2] has become a competitive paradigm in which the platform publishes spatial tasks, and workers arrive at specific locations to execute them. Many applications benefit from combining with SC, such as Waze and Uber [2], which help with data collection and urban services.

Most existing studies concentrate on allocating tasks to suitable workers [3, 4] to maximize the utility of an SC platform. [5] designs a Bisection-based framework in the SC scenario of multiple workers to optimize the travel cost and the number of completed tasks. [6] devises a strategy to maximize the total

weighted completion quality of all tasks under a limited budget. These researches focus on task allocation from the perspective of the platform while ignoring the entitlement of workers. In such a case, selfish and rational workers might be unwilling to execute tasks, leading the platform to fail to achieve the desired effect in practical application. Motivated by this, we focus on the task selection from the perspective of workers who arrive at an SC system online, in which there are two main challenges. First, since the task information is unknown to workers, it is essential to adopt suitable techniques to learn from historical tasks of the same categories. Second, it is challenging to design collaboration to benefit each worker since workers are selfish and rational.

Multi-Armed Bandit (MAB) [7] is an efficient learning technique to deal with information uncertainty and performs well in crowdsourcing scenarios [8]. Multi-Agent Multi-Armed Bandit (MAMAB) [9] is an extension of the traditional MAB, which assists multiple agents in learning the information and taking action to maximize the individual reward [10]. It has been introduced to some application scenarios, such as wireless caching [11]. In this paper, the problem of multiple workers deciding on task selection can be modeled as a MAMAB problem. Note that our optimization goal is to maximize the individual profit for each worker, which is somewhat different from the goal of the traditional MAMAB problem (maximize the overall reward).

Compared to the commonly used assumption [12, 13] that the payment for executing tasks follows a specific distribution, we consider a more practical situation in which the payment is determined by both the task completion and the worker contribution. Since the platform often gives no more information except payment, communication among workers is essential to learn task information. Based on the development and popularity of social networks, workers can collaborate on collecting and processing task information, which helps update their knowledge about tasks of different categories.

In an SC system, selfish and rational workers who arrive online want to select suitable tasks to maximize individual profit. We propose a Payment-Estimation-Based Solution (PEBS) to help with task selection. It utilizes the social network to allow workers to collaboratively learn different categories of tasks from the historical data, which helps with the decision on task selection. The idea of probability matching in Thompson Sampling (TS) [14] is employed to build and adjust the probability distribution of task information. The sampled value is an estimation of task information, based on which workers can calculate the profit for executing tasks and make decisions on task selection. The traditional TS technique applies to the Bernoulli MAB, which requires that the reward of each arm follows the Bernoulli distribution. We extend it to our problem, where task information follows a Gaussian distribution, reflecting that some factors may slightly affect tasks of the same category with the same characteristics. PEBS allows workers to estimate profit accurately and select suitable tasks.

We highlight the main contributions as follows. (1) Different from existing studies that focus on task allocation for the platform, we design a task selection mechanism from the perspective of workers, supplementing the missing angle of the previous work. (2) We utilize the social network among workers. Workers

can collaboratively learn from the historical data of different categories of tasks, which helps with task selection. (3) We extend the traditional TS technique to estimate the task information under Gaussian distributions. Workers realize the accurate profit estimation and make appropriate decisions based on it. (4) Simulation results are provided to demonstrate the effectiveness of our proposed mechanism in optimizing the individual profit of workers.

The remainder of the paper is organized as follows. We first describe the model and formulate the optimization goal in Sect. 2. The details of our mechanism and proposed PEBS are in Sect. 3. In Sect. 4, simulations are provided to validate the performance of PEBS. We conclude this paper in Sect. 5.

2 System Model and Problem Formulation

2.1 System Model

The SC system runs in N cycles (usually equal-length time slots) [15] and includes a task requester, an SC platform, and a set of registered workers.

At the beginning of each cycle k , the requester will submit a set of tasks $\mathcal{T}_k = \{t_1, t_2, \dots, t_o\}$ to the SC platform. Each task $t_j \in \mathcal{T}_k$ can be presented as $t_j = \{l_j, c_j, b_j\}$, where l_j is the location of task t_j , c_j denotes the category that task t_j belongs to, b_j is the budget of the platform for it (which is the maximum value that the platform is willing to pay for workers who execute t_j in the cycle). There are m different categories of tasks in the SC system, and each task t_j belongs to one specific category c_j (c_j is an integer that lies in $[1, m]$).

After receiving the task request from the requester, the platform will publish the task set to the registered workers $\mathcal{W} = \{w_1, w_2, \dots, w_n\}$. In each cycle, the registered workers arrive at the system online. Use $w_i = \{a_i^k, l_i^k, r_i^k\}$ to present worker w_i in cycle k , where a_i^k is the time that worker w_i arrives at the system, l_i^k represents the location of worker w_i , r_i^k is the reachable distance of worker w_i . For worker w_i who doesn't participate in the system in cycle k , it has $a_i^k = \infty$. Else, worker w_i participates in the system at time a_i^k with location l_i^k . Considering the spatial constraint, workers can select and execute tasks in the circle reachable region with center l_i^k and radius r_i^k . The set of tasks in the reachable region of worker w_i is denoted as $\mathcal{T}_{i,k}$. Worker w_i can know the number of workers who have selected each task t_j until the time a_i^k in cycle k , and the number is denoted as $s_{i,j}^k$. According to a specific strategy, worker w_i makes the decision $\zeta_{i,k}$ on task selection from the reachable task set $\mathcal{T}_{i,k}$ to maximize the individual profit.

At the end of each cycle, the platform will calculate the completion degree of each task t_j and the contribution of each worker w_i who has selected t_j . The calculation of the completion degree and the contribution is shown in detail in Sect. 2.3. Based on the above calculation results and the budget b_j for each task t_j , the platform gives the payment to each worker w_i .

2.2 Social Network Communication Model

In the multi-worker scenario, the social network among workers effectively allows workers to communicate and share the task information.

We use a dynamic undirected graph $\mathcal{G}_k = \langle V_k, E_k \rangle$ to represent the social network among workers. Each node in V_k is one registered worker, and each edge in E_k represents a communication link between a pair of workers. Workers with communication links are social friends and can share information about different categories of tasks. Before the actual operation of the system, workers have an original social network \mathcal{G}_0 . At the end of each cycle, workers who have selected the same tasks will become social friends since they communicate to get task information. The number of communication links in the dynamic social network has a non-decreasing nature (*i.e.*, $|E_{k-1}| \leq |E_k|, \forall k \in [1, N]$).

The social network helps workers obtain and learn from the historical data of different categories of tasks. At the end of each cycle, the platform gives workers no information except payments. The payment is related to the completion degree of tasks and the contribution of workers. Each worker w_i , who has executed task t_j , only knows the length of its execution duration and the obtained payment. Based on the communication with workers who have selected the same task t_j , worker w_i can calculate its contribution and further deduce the task information, such as the execution difficulty. Since workers and friends may select tasks belonging to different categories, the social network allows them to obtain information about categories of unselected tasks. Workers adopt specific strategies to deal with the information from social friends to get observations on task information. In a word, the social network accelerates the process of workers learning different categories of tasks, which helps with task selection.

2.3 Problem Formulation

Since workers are selfish and rational individuals, each worker wants to select suitable tasks to maximize the individual profit.

The profit of worker w_i in cycle k is the difference between the obtained payment and its reserve price. The payment is related to the completion degree of tasks and the contribution of workers. Next, we formulate these concepts.

The completion degree of tasks generally satisfies the submodularity. It increases at a diminishing amplitude with the increase in the number of workers who execute them [16, 17] and never exceeds 1. It can be formalized as below:

$$c_t(t_j, k) = \left(1 - \frac{1}{(1.5 + e_{j,k})^{|U_{j,k}|}}\right), \quad (1)$$

in which $U_{j,k}$ represents the set of workers who execute task t_j in cycle k and $e_{j,k} \in (0, 1)$ reflects the execution difficulty of task $t_j \in \mathcal{T}_k$. The execution difficulty of tasks is unknown to workers. For tasks belonging to the same category, their execution difficulty follows an independent and identical distribution with an unknown expectation. The expectation reflects the average execution difficulty of a specific category of tasks, and the fluctuations around the expectation reflect other factors that influence task execution.

The contribution of worker w_i to task t_j can be calculated as below:

$$c_w(w_i, t_j, k) = \mathbb{I}_{i,j}^k \cdot \frac{d_{i,k}}{\sum_{w_z \in U_{j,k}} d_{z,k}}. \quad (2)$$

$\mathbb{I}_{i,j}^k$ is a decision variable that returns 1 if worker w_i selects task t_j in cycle k and 0 otherwise. $d_{i,k}$ is the length of duration for worker w_i to execute task t_j from the arrival time a_i^k to the end of the cycle k . Equation (2) is the ratio of the execution duration of w_i to that of all workers who execute the same task. Note that $\sum_{w_z \in \mathcal{U}_{j,k}} c_w(w_z, t_j, k) = 1$.

Based on the calculation of completion degree and contribution, the platform pays worker w_i who has executed task t_j in cycle k , as below:

$$R_{true}(w_i, t_j, k) = \begin{cases} b_j \cdot c_t(t_j, k) \cdot c_w(w_i, t_j, k) & c_t(t_j, k) < \epsilon \\ b_j \cdot c_w(w_i, t_j, k) & c_t(t_j, k) \geq \epsilon \end{cases} \quad (3)$$

where b_j is the budget of the platform for task t_j , $c_t(t_j, k)$ is the completion degree of task t_j and $c_w(w_i, t_j, k)$ represents the contribution of worker w_i . If the completion degree is more than ϵ , the tasks are considered fully completed, and the platform pays all budgets to workers who execute them.

Each worker w_i has an expected minimum payment for executing task t_j in cycle k , called the reserve price $f_{i,j}^k$. It is positively correlated with the budget b_j of task t_j and slightly fluctuates due to the different demands of workers. Workers suffer losses if the obtained payment is less than it. The profit of worker w_i who selects task t_j in cycle k is the difference between the payment and the reserve price. We denote it as $P(w_i, t_j, k) = R_{true}(w_i, t_j, k) - f_{i,j}^k$.

We focus on the task selection problem from the perspective of workers. The goal of each worker $w_i \in \mathcal{W}$ is to select a suitable task $t_j \in \mathcal{T}_k$ in each cycle k and maximize the cumulative profit on N cycles. Combining the concepts defined above, we formulate the optimization goal for worker w_i as follows.

$$\text{Maximize} : \sum_{k=1}^N \sum_{t_j \in \mathcal{T}_{i,k}} \mathbb{I}_{i,j}^k \cdot \left(R_{true}(w_i, t_j, k) - f_{i,j}^k \right) \quad (4)$$

$$\text{Subject to} : \mathcal{T}_{i,k} = \{t_j \mid \rho(l_i^k, l_j) \leq r_i^k, t_j \in \mathcal{T}_k\}, \quad \forall k \in [1, N] \quad (5)$$

$$\sum_{t_j \in \mathcal{T}_{i,k}} \mathbb{I}_{i,j}^k \leq 1, \quad \forall k \in [1, N] \quad (6)$$

$$\sum_{t_j \in \mathcal{T}_k} s_{z,j}^k \leq \sum_{t_j \in \mathcal{T}_k} s_{i,j}^k, \quad \forall w_z \in \{w_z \mid a_z^k \leq a_i^k, w_z \in \mathcal{W}\}, k \in [1, N] \quad (7)$$

$\mathbb{I}_{i,j}^k$ returns 1 if worker w_i selects task t_j in cycle k and 0 otherwise. $\rho(l_i^k, l_j)$ calculates the distance between worker w_i and task t_j . Constraint (5) indicates that workers cannot select spatial tasks out of their reachable region; (6) indicates that workers select no more than one task per cycle; and (7) means that workers make decisions according to the arrival order and execute the selected tasks until the end of the cycle. Let online workers act as agents, and tasks act as arms. We model the decision-making problem of multiple workers as a MAMAB problem. The workers face the dilemma between exploring and exploiting tasks whose completion degree is determined by the prior unknown parameter in each cycle to maximize the cumulative individual profit.

3 Mechanism Design

3.1 Social-Network-Based Observation

At the beginning of each cycle k , the platform publishes a set of tasks \mathcal{T}_k . Since the execution difficulty follows an independent and identical distribution for tasks in the same category, workers can estimate the execution difficulty of each task $t_j \in \mathcal{T}_k$ according to the historical data of tasks in the same categories.

Worker w_i uses $\hat{X}_i^{1 \times m}(k)$, $\hat{Y}_i^{1 \times m}(k)$ to represent its estimation of the cumulative execution difficulty and the total execution times of tasks in m categories till cycle k . Worker w_i can get an observation of the execution difficulty for all categories of tasks in cycle k , denoted as $\hat{e}_i^{1 \times m}(k) = \{\hat{e}_{i,1}(k), \hat{e}_{i,2}(k), \dots, \hat{e}_{i,m}(k)\}$ ($\hat{e}_{i,c_j}(k) = \hat{X}_{i,c_j}(k)/\hat{Y}_{i,c_j}(k)$). Then, we take $\hat{X}^{n \times m}(k)$ and $\hat{Y}^{n \times m}(k)$ to represent all workers' estimation of the cumulative execution difficulty and the total execution times of m task categories till cycle k .

With the continuous execution of tasks and information communication, worker w_i will update $\hat{X}(k)$ and $\hat{Y}(k)$ through two processes: the individual update based on the information deduced from the payment for executing task $\zeta_{i,k}$ and the collaborative update with friends based on the social network.

The individual update is as follows:

$$\hat{X}_i(k) = \hat{X}_i(k) + \alpha_i(k), \quad (8)$$

$$\hat{Y}_i(k) = \hat{Y}_i(k) + \beta_i(k). \quad (9)$$

$\beta_i^{1 \times m}(k)$ reflects the category of the task selected by worker w_i in cycle k , and $\alpha_i^{1 \times m}(k)$ reflects the execution difficulty inferred from the obtained payment. If worker w_i selects task t_j in cycle k , we have $\beta_{i,c_j}(k) = 1$ and $\alpha_{i,c_j}(k) = e_{j,k}$. The other elements in $\beta_i(k)$ and $\alpha_i(k)$ are equal to 0.

The collaborative update based on the social network is as follows:

$$\hat{X}(k) = H_k \times \hat{X}(k-1), \quad (10)$$

$$\hat{Y}(k) = H_k \times \hat{Y}(k-1). \quad (11)$$

$H_k = I_n - \lambda_k \cdot (D_k - G_k)$ is a matrix assisting in updating $\hat{X}(k)$ and $\hat{Y}(k)$. I_n is the identity matrix. λ_k is the parameter that reflects the influence of friends on collaborative update. D_k is the degree matrix of \mathcal{G}_k , and G_k is the adjacency matrix. H_k adopts the idea of weighted summation and the sum of each row of H_k is 1. λ_k lies in $(0, 1/(\max\{D_k[i][i] \mid 1 \leq i \leq n\} - 1))$ to guarantee elements in H_k non-negative. We can modify λ_k to adjust the dependence of workers on friends in cycle k . The above updates allow workers to get information about multiple task categories and get an accurate observation of execution difficulty.

For m task categories, we use $\hat{P}^{n \times m}(k)$ and $\hat{Q}^{n \times m}(k)$ to represent all n workers' estimation of the cumulative length of execution duration and the total number of workers who have selected them. The individual update is as follows:

$$\hat{P}_i(k) = \hat{P}_i(k) + \gamma_i(k), \quad (12)$$

$$\hat{Q}_i(k) = \hat{Q}_i(k) + \delta_i(k). \quad (13)$$

If worker w_i selects task t_j in cycle k , we have $\gamma_{i,c_j}(k) = \sum_{w_z \in U_{j,k}} d_{z,k}$ and $\delta_{i,c_j}(k) = |U_{j,k}|$. The other elements in $\gamma_i^{1 \times m}(k)$ and $\delta_i^{1 \times m}(k)$ are equal to 0.

The cooperative update on $\hat{P}(k)$ and $\hat{Q}(k)$ is as follows:

$$\hat{P}(k) = H_k \times \hat{P}(k-1), \quad (14)$$

$$\hat{Q}(k) = H_k \times \hat{Q}(k-1). \quad (15)$$

Based on $\hat{P}(k)$ and $\hat{Q}(k)$, workers can get an observation of the length of duration for one worker to execute a certain category c_j of tasks (*i.e.*, $\hat{P}_{i,c_j}(k)/\hat{Q}_{i,c_j}(k)$).

3.2 Thompson-Sampling-Based Payment Estimation

For each task $t_j \in \mathcal{T}_k$, worker $w_i \in \mathcal{W}$ needs to estimate its execution difficulty $\tilde{e}_{i,j}^k$ in cycle k based on the historical observations on its category.

Based on task execution and social-network-based communication, workers get historical observations about the execution difficulty of different categories of tasks. Then, workers adopt the probability matching idea in Thompson Sampling to estimate the execution difficulty. Specifically, workers generate probability models based on historical observations and then sample from the generated probability density curves as the estimated execution difficulties of tasks.

We use the normal distribution to model the execution difficulty (which can be easily adjusted to other types of distributions). Let μ_{i,c_j}^k and σ_{i,c_j}^k represent the mean and standard deviation of the distribution that worker w_i generates for the category c_j . To accurately estimate the execution difficulty of t_j , worker w_i constantly adjusts the two parameters based on the observation of the execution difficulty $\hat{e}_{i,c_j}(k)$ of its category, which is as follows:

$$\mu_{i,c_j}^k = \frac{\tau_{i,c_j}^k \mu_{i,c_j}^k + \mathbb{I}_{i,c_j}^k \cdot \tau \hat{e}_{i,c_j}(k)}{\tau_{i,c_j}^k + \mathbb{I}_{i,c_j}^k \cdot \tau}, \quad (16)$$

$$\tau_{i,c_j}^k = \tau_{i,c_j}^k + \mathbb{I}_{i,c_j}^k \cdot \tau. \quad (17)$$

\mathbb{I}_{i,c_j}^k is a decision variable, where $\mathbb{I}_{i,c_j}^k = 1$ means that there is a new observation on the execution difficulty of the task category c_j in cycle k ; otherwise, $\mathbb{I}_{i,c_j}^k = 0$. τ is a worker-defined parameter to control the rate at which the probability density curve changes. We calculate the standard deviation by $\sigma_{i,c_j}^k = \sqrt{1/\tau_{i,c_j}^k}$. The sampled value from the execution difficulty distribution of task category c_j is an estimation of execution difficulty of t_j in cycle k , denoted as $\tilde{e}_{i,j}^k$. With the obtain of observations, the standard deviation σ_{i,c_j}^k decreases, and the estimated execution difficulty $\tilde{e}_{i,j}^k$ gets closer to the mean of the distribution.

Worker w_i estimates the payment for executing task t_j in cycle k according to the task budget b_j and the estimated contribution, as below:

$$R_{est}(w_i, t_j, k) = b_j \cdot \frac{d_{i,k}}{\max(s_{i,j}^k, \tilde{u}_{i,j}^k) \cdot \hat{P}_{i,c_j}(k)/\hat{Q}_{i,c_j}(k)}. \quad (18)$$

Algorithm 1 Payment-Estimation-Based Solution (PEBS) for Task Selection

Require: registered worker set \mathcal{W} , cycle number N and original social network \mathcal{G}_0

- 1: **for** $k = 1 : N$ **do**
- 2: The platform publishes a task set \mathcal{T}_k .
- 3: $\mathbb{W}(k) = \text{sort}(\{w_i \mid a_i^k \neq \infty, w_i \in \mathcal{W}\}, \text{key} = a_i^k)$.
- 4: **for** $w_i \in \mathbb{W}(k)$ **do**
- 5: $\mathbb{T} \leftarrow []$.
- 6: **for** $t_j \in \mathcal{T}_k$ **do**
- 7: Update μ_{i,c_j}^k and σ_{i,c_j}^k for its category c_j .
- 8: Estimate its execution difficulty $\tilde{e}_{i,j}^k$ by sampling from $N(\mu_{i,c_j}^k, (\sigma_{i,c_j}^k)^2)$.
- 9: **if** $(\hat{Y}_{i,c_j}(k) = 0) \wedge (t_j \in \mathcal{T}_{i,k})$ **then**
- 10: $\mathbb{T} \leftarrow \mathbb{T} + [t_j]$.
- 11: **if** $|\mathbb{T}| > 0$ **then**
- 12: $\zeta_{i,k} = \text{choice}(\mathbb{T})$.
- 13: **else**
- 14: $\mathcal{T}'_{i,k} = \{t_j \mid R_{est}(w_i, t_j, k) - f_{i,j}^k \geq 0, t_j \in \mathcal{T}_{i,k}\}$.
- 15: $\zeta_{i,k} = \text{argmax}_{t_j \in \mathcal{T}'_{i,k}} (R_{est}(w_i, t_j, k) - f_{i,j}^k)$.
- 16: The platform gives payments to workers according to Eq. (3).
- 17: \mathcal{G}_{k-1} updates to \mathcal{G}_k .
- 18: Each worker w_i updates $\hat{X}_i(k)$, $\hat{Y}_i(k)$, $\hat{P}_i(k)$ and $\hat{Q}_i(k)$ individually.
- 19: Workers make collaborative updates to $\hat{X}(k)$, $\hat{Y}(k)$, $\hat{P}(k)$ and $\hat{Q}(k)$.

In practical scenarios, the platform always recruits enough workers to guarantee that the completion degree of most tasks reaches ϵ . Workers reasonably believe that the platform pays all budgets b_j for each task t_j . $d_{i,k}$ is the length of execution duration of worker w_i in cycle k . For tasks in category c_j , $\hat{P}_{i,c_j}(k)/\hat{Q}_{i,c_j}(k)$ is the latest observation of the average length of their execution duration. $s_{i,j}^k$ is the number of workers who have selected task t_j before time a_i^k , and $\tilde{u}_{i,j}^k$ is the minimum needed number of workers for task t_j to be fully completed based on the estimated execution difficulty $\tilde{e}_{i,j}^k$ ($\tilde{u}_{i,j}^k = \min\{z \mid 1/(1.5 + \tilde{e}_{i,j}^k)^z \leq 1 - \epsilon, z \in \mathbb{Z}\}$). We take the larger of the two as the estimated number of workers who select task t_j in cycle k to deal with the scenario registered workers arrive online.

3.3 Detailed Algorithm

According to the above discussion, we propose the Payment-Estimation-Based Solution (PEBS) for task selection of workers as shown in Algorithm 1.

At the beginning of each cycle, the platform publishes a task set \mathcal{T}_k (Line 2). Workers who arrive at the system online make decisions on task selection according to the arrival order (Lines 3–14). Based on the historical observations, workers generate probability models of execution difficulty and estimate the execution difficulty for each task by sampling (Lines 7–8). If there are reachable tasks that worker w_i has no knowledge of the execution difficulty of their

categories, worker w_i randomly selects one of them (Line 12). Else, worker w_i selects the reachable task with the maximum and non-negative estimated profit (Lines 14–15). The social network updates at the end of each cycle (Line 17). Based on the social network, workers learn from the obtained payment and then get accurate observations on the information of multiple task categories (Lines 18–19).

4 Performance Evaluation

4.1 Settings, Metrics and Baselines

The performance of our proposed mechanism in maximizing the individual profit of workers is evaluated in a scenario where the number of cycles is 400. In each cycle, the platform publishes 100 tasks, and the number of online workers participating in task selection is 600. The reachable region of each worker is large enough to cover all tasks. The platform pays all the budgets for tasks whose completion degree exceeds $\epsilon = 0.9$. The original social network \mathcal{G}_0 is generated randomly. We set $\lambda_k = 1/|\mathcal{W}|$, which is a constant and prevents the frequent adjusting of the collaborative update caused by the dynamic social network.

The ratio of workers who have non-negative profit in all registered workers (hereafter referred to as the target ratio) and the profit of each worker are the key metrics to evaluate the performance of task selection algorithms. Since advanced strategies designed for specific scenarios are not suitable for solving our problem, we introduce the baselines that adopt random and greedy ideas. The random strategy allows workers to select tasks randomly. The greedy strategy allows task selection based on the estimated payment ($\zeta_{i,k} = \operatorname{argmax}_{t_j \in \mathcal{T}_{i,k}} (b_j \cdot 1/(1 + s_{i,j}^k))$). Since worker w_i has no information about task execution without the social network, the payment estimation here can only utilize the number of workers who have selected each task $t_j \in \mathcal{T}_k$ till time a_i^k in cycle k .

4.2 Simulation Results

We first illustrate the simulation results regarding the target ratio. In Fig. 1, the target ratio based on random and greedy fluctuates mainly in (0.65, 0.70) and (0.60, 0.65), respectively. It fails to optimize because these two strategies without the social network cannot help workers learn from the payment. The target ratio based on PEBS rises rapidly and mainly fluctuates slightly in (0.9, 1.0), reflecting its effectiveness in payment estimation and task selection. To verify the effectiveness of the social network, we introduce a simplified version of PEBS without social-network-based communication, named PEBS-NS. We assume that the platform tells each worker the arrival time of all workers and the execution information of tasks they select in each cycle. The target ratio rises after about 100 cycles under such an ideal and unrealistic assumption, demonstrating the advantages and practical significance of adopting the social network in PEBS.

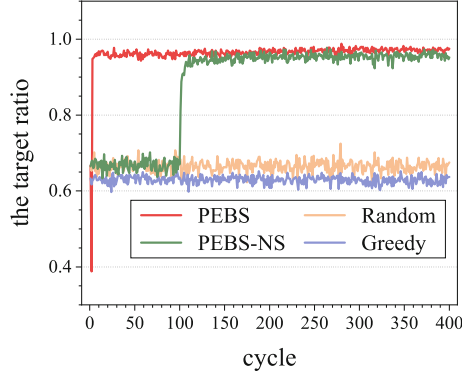


Fig. 1. The target ratio under different task selection strategies.

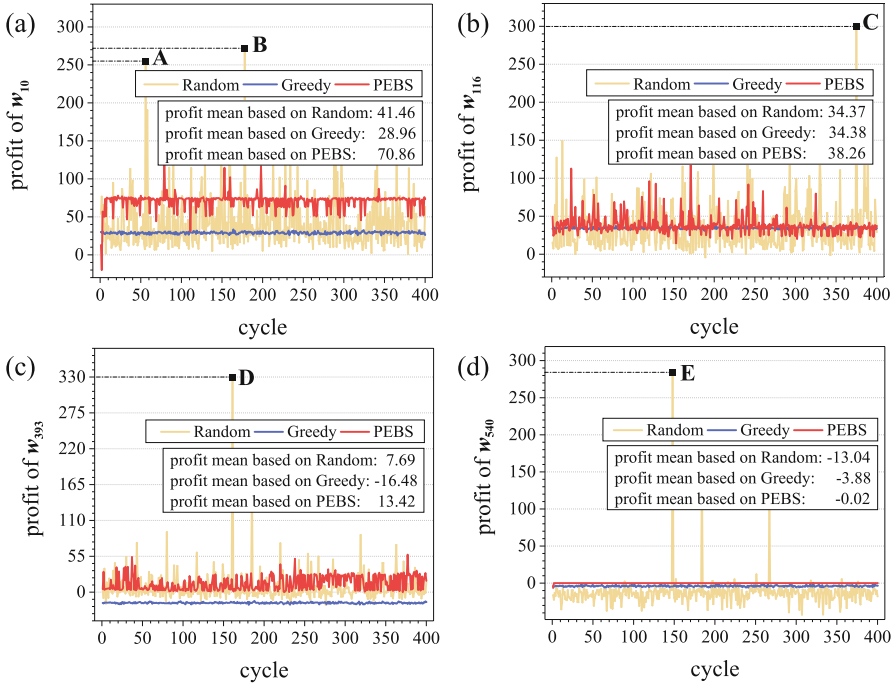


Fig. 2. The profit of workers w_{10} , w_{116} , w_{393} and w_{540} in each cycle.

Let the indexes of workers reflect their arrival order in each cycle (*i.e.*, w_1 is always the first one arriving at the system, w_2 is the second, *etc.*). We first select four workers to investigate the profit of workers in each cycle and then study the impact of the arrival order on the profit of workers. In Fig. 2, the random strategy brings strong fluctuations to the profit of the four workers, while the greedy and PEBS strategies stabilize it. It is reasonable because the latter two

strategies consider the number of workers who have selected each task t_j , which is stable because of the fixed arrival order. In Fig. 2(d), only PEBS helps w_{540} avoid negative profit effectively because it can assist in accurate payment estimation. The standard deviations of their profit in all cycles are listed in Table 1, which confirms the observation result in Fig. 2. Table 1 also shows the profit information of all workers in all cycles under different strategies. The greedy strategy brings stable but lowest profit. The random strategy brings workers more profit, but workers take a risk because of the highest standard deviation. Compared with them, PEBS can guarantee the highest profit and lower decision-making risk owing to the relatively low standard deviation.

Table 1. Profit information under different strategies.

Strategy	w_{10}	w_{116}	w_{393}	w_{540}	All workers	
	Standard deviation				Standard deviation	Mean
Random	34.25	31.22	23.93	15.53	22.75	14.61
Greedy	1.45	1.13	1.04	0.69	1.13	10.12
PEBS	10.38	12.80	11.02	0.49	10.10	20.21

We then discuss the cases where the random strategy leads to abnormally high profits, marked in Fig. 2. Their relevant information is shown in Table 2. The tasks related to these cases all have considerable budgets. At the same time, the completion degree of these tasks and the contribution of the workers are both high. The emergence of abnormally high profit highly depends on task budgets and the decisions of other workers, which cannot be controlled and replicated.

Table 2. Information on marked cases.

Case	Task information			Worker situation	
	Index	Budget	Completion degree	Contribution	Profit
A	61	498	90.77%	61.14%	254.99
B	61	498	95.40%	64.58%	271.97
C	11	495	77.43%	90.97%	299.71
D	29	495	86.34%	88.42%	330.02
E	11	495	88.08%	76.74%	284.22

Figure 3 shows the average profit of each worker in all cycles. As the index of workers increases, their duration for executing tasks becomes short (which always leads to a lower contribution), and the average profit based on all strategies tends to decline. The profit of later arriving workers who select tasks based on PEBS may be less than those who select tasks based on other strategies. It is intuitive

because budgets for tasks are limited. If workers arriving earlier get paid more, the profit of other workers will be less accordingly. PEBS avoids workers getting negative profits and helps the worker community achieve a higher overall profit.

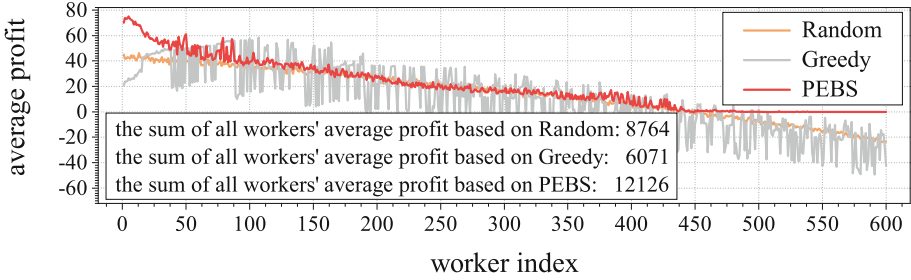


Fig. 3. Average profit per worker in all cycles.

5 Conclusion

This paper proposes a mechanism to deal with the task selection problem from the perspective of online workers in spatial crowdsourcing. Since we consider a scenario where the information of different categories of tasks is unknown, we model the problem as a Multi-Agent Multi-Armed Bandit problem and propose the PEBS algorithm to deal with it. PEBS allows workers to learn from the historical information of different categories of tasks through the social network. It adopts the probability matching idea in Thompson Sampling to help estimate the difficulty and the profit of executing different tasks in the current cycle. Extensive simulations verify the significant performance and stability of PEBS in guaranteeing the individual profit of workers.

Acknowledgement. This research was supported in part by the National Natural Science Foundation of China (Grant No. U20A20182, 62102275, 61873177, 62072322), in part by the NSF of Jiangsu in China under Grant BK20210704, in part by the NSF of the Jiangsu Higher Education Institutions of China under Grant 21KJB520025.

References

1. Zhao, Y., Han, Q.: Spatial crowdsourcing: current state and future directions. *IEEE Commun. Mag.* **54**(7), 102–107 (2016)
2. Tong, Y., Chen, L., Shahabi, C.: Spatial crowdsourcing: challenges, techniques, and applications. *Proc. VLDB Endow.* **10**(12), 1988–1991 (2017)
3. Xu, W., Huang, H., Sun, Y.E., Li, F., Zhu, Y.: DATA: a double auction based task assignment mechanism in crowdsourcing systems. In: *CHINACOM*, pp. 172–177 (2013)
4. Huang, H., Xin, Y., Sun, Y.E., Yang, W.: A truthful double auction mechanism for crowdsensing systems with max-min fairness. In: *IEEE WCNC*, pp. 1–6 (2017)

5. Deng, D., Shahabi, C., Zhu, L.: Task matching and scheduling for multiple workers in spatial crowdsourcing. In: ACM SIGSPATIAL, pp. 1–10 (2015)
6. Gao, G., Huang, H., Xiao, M., Wu, J., Sun, Y.E., Du, Y.: Budgeted unknown worker recruitment for heterogeneous crowdsensing using CMAB. *IEEE Trans. Mob. Comput.* (2021)
7. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.* **47**(2), 235–256 (2002)
8. Gao, X., Chen, S., Chen, G.: MAB-based reinforced worker selection framework for budgeted spatial crowdsensing. *IEEE Trans. Knowl. Data Eng.* **34**(3), 1303–1316 (2022)
9. Shahrampour, S., Rakhlin, A., Jadbabaie, A.: Multi-armed bandits in multi-agent networks. In: *IEEE ICASSP*, pp. 2786–2790 (2017)
10. Landgren, P., Srivastava, V., Leonard, N.E.: Distributed cooperative decision making in multi-agent multi-armed bandits. *Automatica* **125**, 109445 (2021)
11. Xu, X., Tao, M., Shen, C.: Collaborative multi-agent multi-armed bandit learning for small-cell caching. *IEEE Trans. Wireless Commun.* **19**(4), 2570–2585 (2020)
12. Pu, L., Chen, X., Xu, J., Fu, X.: Crowdlet: optimal worker recruitment for self-organized mobile crowdsourcing. In: *IEEE INFOCOM*, pp. 1–9 (2016)
13. Xiao, M., Wu, J., Huang, L., Wang, Y., Liu, C.: Multi-task assignment for crowdsensing in mobile social networks. In: *IEEE INFOCOM*, pp. 2227–2235 (2015)
14. Chapelle, O., Li, L.: An empirical evaluation of Thompson sampling. In: *Advances in Neural Information Processing Systems*, vol. 24 (2011)
15. Du, Y., et al.: Bayesian co-clustering truth discovery for mobile crowd sensing systems. *IEEE Trans. Industr. Inf.* **16**(2), 1045–1057 (2020)
16. Lu, Z., Wang, Y., Li, Y., Tong, X., Mu, C., Yu, C.: Data-driven many-objective crowd worker selection for mobile crowdsourcing in industrial IoT. *IEEE Trans. Industr. Inform.* (2021)
17. Qu, G., Brown, D., Li, N.: Distributed greedy algorithm for multi-agent task assignment problem with submodular utility functions. *Automatica* **105**, 206–215 (2019)



Privacy-Aware Task Allocation Based on Deep Reinforcement Learning for Mobile Crowdsensing

Mingchuan Yang, Jinghua Zhu^(✉), Heran Xi, and Yue Yang

School of Computer Science and Technology, Heilongjiang University,
Harbin 150080, China
zhujinghua@hlju.edu.cn

Abstract. Mobile crowdsensing (MCS) is a new paradigm for data collection, data mining and intelligent decision-making using large-scale mobile devices. The efficient task allocation method is the key to the high performance of MCS. The traditional greedy algorithm or ant algorithm assumes that workers and tasks are fixed, which is not suitable for the situation where the location and quantity of workers and tasks change dynamically. Moreover, the existing task allocation methods usually collect the information of workers and tasks by the central server for decision-making, which is easy to lead to leakage of workers' privacy. In this paper, we propose a task allocation method with privacy protection using deep reinforcement learning (DRL). Firstly, the task allocation is modeled as a dynamic programming problem of multi-objective optimization, which aims to maximize the benefits of workers and platform. Secondly, we use DRL for training and learning model parameters. Finally, the local differential privacy method is used to add random noise to the sensitive information, and the central server trains the whole model to obtain the optimal allocation strategy. The experimental results on the simulated data set show that compared with the traditional methods and other DRL based methods, our proposed method has significantly improved in different evaluation metrics, and can protect the privacy of workers.

Keywords: Mobile crowdsensing · Task allocation · Deep reinforcement learning · Differential privacy

1 Introduction

MCS, first proposed by Ganti et al. [4], is a new method for sensing and sharing data among users or communities, which has developed and penetrated into various fields such as smart medical care and smart cities, requiring a large number of data sets and users. As is shown in Fig. 1, the task allocation system of MCS is mainly composed of three parts: platform, workers and tasks. Tasks are assigned to workers by the platform based on a certain revenue calculation mechanism, and then workers collect relevant perceptual data and upload it to the platform to obtain the corresponding revenue. On the one hand, tasks should

be reasonably allocated to appropriate workers with the maximization of total platform profit and the benefits of selected workers. On the other hand, workers usually cannot avoid revealing their own private information when uploading information. Based on the above two aspects, the reasonable task allocation mechanism and the privacy protection of worker information are particularly important. The traditional task allocation algorithms, such as greedy algorithm [12] and ant algorithm [2], are only suitable for small-scale data sets in static systems with fixed workers and tasks information. Therefore, the algorithms of DRL are introduced into such dynamic systems by more and more researchers to solve the corresponding dynamic programming problems.

DRL can interact with the environment through the agent's choice of actions and obtain the corresponding state and reward based on past experience [8]. In dynamic MCS problems, DRL can usually obtain a better performance. Among the methods of DRL, DQN [1] and A3C [9] can perform well but only in discrete action spaces. DDPG [10] is an offline and deterministic method, which is unsuitable for dynamic scenarios requiring real-time control solutions. PPO is a model-free, strategy-based, and gradient-based method that performs extremely well in continuous control problems [7]. In related works, the PPO framework even works in composite state and action sets and has excellent performance and faster convergence than other method of DRL. So we adopt the PPO framework in this paper.

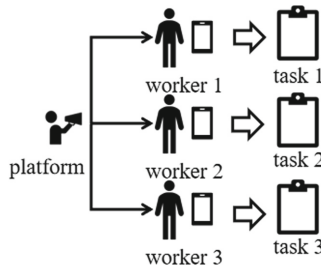


Fig. 1. Diagram of MCS task assignment system

The task allocation problem in dynamic environment is modeled as an optimization problem of dynamic programming based on discrete data set, which we use the methods of DRL and differential privacy to solve. Specifically, at the beginning of each iteration, the allocation strategies and benefits of platform, worker benefits, existing task information and worker information in the previous iteration are observed. According to the observation results, the tasks assigned by workers and the order of tasks will be determined according to DRL. In this process, differential privacy [3] is used to blur the privacy information related to workers. Our objective is to maximize the total income of the platform and ensure the maximum income of workers. It is defined as the joint constraint between workers' income and the total income of the platform. The main contributions of this paper are summarized as follows:

- We model the task assignment problem in the MCS dynamic scene as a multi-objective optimization problem and prove it to be a NP-hard problem. The dynamic system of changing worker-task status information and the need for privacy protection of worker-platform data interaction is considered in this problem.
- We present a method based on DRL and local differential privacy to solve this optimization problem. Our method based on DRL is more suitable to solve such dynamic and non-deterministic MCS problems.
- Through experiments, The results show that our model has stable performance and good convergence. In addition, ablation experiments are carried out to verify the effectiveness of adding privacy protection methods.

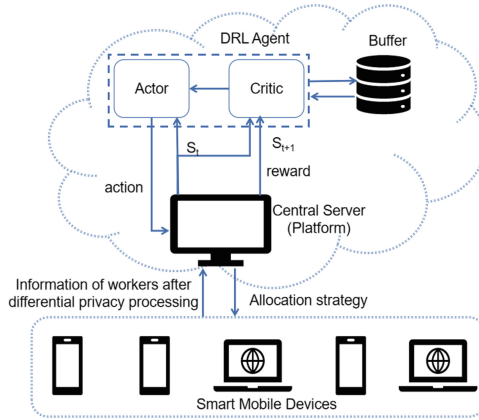


Fig. 2. Illustration of the system model.

2 Problem Description

We assume that there are n workers $W = \{w_1, w_2, \dots, w_n\}$ carrying smart mobile devices in the system, and m tasks $V = \{v_1, v_2, \dots, v_m\}$. The i -th worker is defined by $w_i = \{P_{w_i}, V_{w_i}, x_i, y_i\}$, and the j -th task is represented by $v_j = \{t_{v_j}, r_{v_j}, x_j, y_j, r_{p_j}\}$, where (x, y) represents the location coordinates. P_{w_i} is the cost set of all tasks of the worker i . V_{w_i} is the task queue that the worker is currently assigned. t_{v_j} represents the time required for the task j . r_{v_j} represents the reward of the task j , and r_{p_j} represents the profit that the platform can obtain from this task. The system is a dynamic environment, that is, the status and location information of workers and tasks changes continuously. Workers will become “idle” after completing existing tasks. At this time, it is necessary to put the “idle” workers in the queue of “to-be-assigned” workers. And the tasks that each worker can accept are also limited, which needs to be related to the worker’s compensation and the constraints of task on the worker’s revenue.

Considering the difference in workers’ costs, the task cost should be different for each worker. Each worker calculates the task cost according to the distance

and the amount of unfinished tasks. Here, the cost of the worker i for the task j is defined as follows:

$$P_{w_i}^j = \zeta \sqrt{(x_i - x_j)^2 - (y_i - y_j)^2} + \alpha \theta_i \quad (1)$$

$$\theta_i = \sum_{v_j \in V_{w_i}} t_{v_j} \quad (2)$$

where θ_i represents the total time required for the worker to complete the task. ζ represents the Euclidean distance weight from the worker i to the j -th task point, and α is the time weight. Equation (1) reflects the difference in the task cost of each worker, while the compensation of each task for all workers is set to a certain constant.

Therefore, the revenue of worker i and the total profit of the platform can be defined as:

$$r_i = \sum_{v_j \in V_{w_i}} (r_{v_j} - P_{w_i}^j) \quad (3)$$

$$R_p = \sum_{v_j \in V_{out}} (r_{p_j} - r_{v_j}) \quad (4)$$

where r_i represents the revenue of the i -th worker, and R_p represents the total profit of the platform at this time. V_{out} represents the set of all assigned tasks at this time. r_{p_j} , as described above, is the profit obtained by the platform when the task is completed.

Then, the dynamic policy optimization problem based on discrete datasets is defined as:

$$\max. \quad \lambda_1 R_p + \lambda_2 r_i \quad (5)$$

$$\text{s.t.} \quad 0 < P_{\min} \leq P_{w_i}^j, \forall w_i \in W \quad (6)$$

where the objective Eq. (5) represents maximizing the total profit of the platform while also maximizing the revenue of the current worker i . λ_1 is the weight of profit of platform, while λ_2 is the weight of revenue of worker. The constraint in Eq. (6) represents that the reward must be guaranteed not to be less than the minimum value P_{\min} when worker i completes the task j .

From Eqs. (3) and (4), it can be seen that definition of the platform's total profit is negatively correlated with the workers' revenue. The use of the joint optimization method balances the interests of both parties by adjusting the weights and realizes the Nash equilibrium.

The above problem definition proves that the task assignment problem of MCS is a NP-hard problem. First, assume a special case where there is only one worker and the set of tasks does not change. Then, this worker has a maximum travel distance and the revenue paid to the worker is set to zero. Finally, the total profit of the platform is equal to the worker's reward for completing the task, which also maps to the directional motion problem. So it can be inferred that the problem in this article is also a NP-hard problem, while the directional motion problem has been proven to be a NP-hard problem [5].

3 System Model and Algorithm

3.1 Overview of System Model

Our proposed model is shown in Fig. 2. First, each worker’s smart mobile device transmits the relevant information to the central server after differential privacy processing. Then, the central server obtains the global status information of workers and tasks at that moment, formulates the corresponding allocation strategy through the DRL-based dynamic strategy optimization algorithm, and finally transmits it to each worker’s smart mobile device. During the interaction process, the PPO algorithm is used in the central server for training and decision-making. In each round of training, the dynamics and differences of the system are considered, that is, different workers for the same task, will have different cost due to the distance and the amount of unfinished tasks. In each iteration, there may be “idle” worker that has completed the current task. The above problems are considered in the model design, and the global information is dynamically updated in each round of iterations. We use a local differential privacy algorithm, which adds Laplace noise to the process of information interaction between workers and platforms to protect privacy. Finally the globally optimal strategy is obtained.

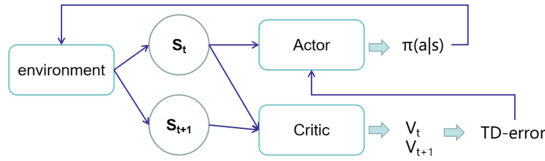


Fig. 3. Block diagram of A-C network.

3.2 PPO

Our model adopts PPO algorithm, which is derived from the idea of A-C network. As shown in Fig. 3, it consists of an actor network and a critic network. At each iteration, actor network selects an action according to a certain action decision probability distribution, and interacts with the environment. Then we can obtain the state and corresponding reward. After that, critic network calculates the corresponding reward function (sometimes TD-error, the weight of evaluation action strategy) and passes it to the actor network and the environment. Then actor network adjusts the decision probability distribution of the action based on this and selects the next action. The optimal strategy is finally obtained when the iteration ends.

For the definition of the reward function of PPO, double constraints are adopted. The action space, state space and reward are defined as follows:

Action Space: The action set contains the matching information between workers and tasks, which is represented in the form of a two-dimensional vector table.

A record of task assignment and task completion sequence will be stored in each worker device. c_k^i is task set assigned to the i -th worker in the k -th iteration. Central server makes the task allocation in each iteration according to the global information obtained from the previous round of interaction. Its definition is as follows:

$$a_k = [c_k^1, \dots, c_k^i, \dots, c_k^n] \quad (7)$$

State Space: The related information of workers and tasks is recorded. It can be represented by the set of available workers W_k and the set of available tasks V_k . At the beginning of each iteration, each worker calculates the cost and benefit according to Eqs. (1) and (3), and interacts with the central server to update the information. The state set is defined as:

$$\mathbf{s}_k = \{W_k, V_k\} \quad (8)$$

Reward: For the definition of reward, we can refer to the joint constraint of (5), which is a Nash equilibrium problem of non-cooperative competition between platform and workers. Here the reward constraint is defined as:

$$\mathbf{r}_k = \lambda_1 R_p + \lambda_2 \sum_{w_i \in W_{\text{out}} \cup W_k} r_{w_i} \quad (9)$$

where W_{out} is the set of workers assigned, and W_k represents the set of workers available. Equation (9) means the balance between platform profit and workers revenue.

During the training of our model, critic network calculates its value and advantage function A_k based on the reward r_k and set of action and state (a_k, s_k) for the evaluation action strategy, and then adjusts the policy π of action selection in the next actor network, which is defined as follows:

$$\text{Value}_\pi(s_k) = \sum_{a \in A} \pi(a_k, s_k) [r_k + \gamma \sum_{s_k \in S} P_{s_k s_{k+1}} \text{Value}_\pi(s_{k+1})] \quad (10)$$

$$A_k = \sum_{i=0}^{\infty} (\gamma \lambda)^i \partial_{k+i} \quad (11)$$

$$\text{Loss} = \hat{E}_k \left[(A_k + \text{Value}_\pi(s_k) - \text{Value}_{\pi_{\text{old}}}(s_k))^2 \right] \quad (12)$$

where the value function for the k -th theory iteration is Value_k , and γ is the discount rate, and $P_{s_k s_{k+1}}$ is the state transition probability. A_k is the advantage function, and λ is a function weight, and $\partial_k = r_k + \gamma \text{Value}_\pi(s_{k+1}) - \text{Value}_\pi(s_k)$. The final loss function of our model can be expressed as *Loss*.

3.3 Task Allocation Algorithm Based on PPO

PPO is used to train and learn a task assignment policy, which strikes a balance between ease of implementation, sample complexity, and ease of tuning, trying

to compute updates to minimize the objective function while ensuring relatively small deviations from previous policies. Therefore, in this paper, the policy optimization process of the DRL agent adopts the PPO algorithm. It includes a historical strategy buffer D , strategy π , value function $Value(a, s)$, weight of actor network θ_a , weight of critic network θ_c and advantage function $A(a, s)$. The pseudo code of the model is shown in Algorithm 1.

First, the PPO is randomly initialized with θ_a and θ_c . The θ_a is used as the initial policy parameter (line 1). Then the iteration start (line 2). After obtaining the information of the current set of available workers W_k and the set of available tasks V_k in the environment, where the location information of the workers has been fuzzed, we get the state of the k -th iteration (lines 3–8). Then, based on the state s_k , actor network selects actions according to the current strategy (line 9). PPO inputs the action set a_k at this time into the environment to calculate the corresponding reward r_k and the state set s_{k+1} in the next round (line 10). A_k and $Value_k$ are calculated in the critic network, and the set $\{s_{k+1}, s_k, a_k, r_k, A_k, Value_k\}$ is stored in D (lines 11–12). When the history policy buffer D is full, the partial derivatives are calculated. Then the policy parameter θ_a is updated based on $\delta\theta_a$ according to the gradient ascent policy (lines 13–15). After learning information from D , the new parameters of the actor network θ_a are assigned to the policy for the next sampling. At the same time, the historical strategy buffer is emptied (line 16).

Algorithm 1. Task Allocation Algorithm Based On PPO

Data: historical strategy buffer D , strategy π , weights of actor-critic network θ_a, θ_c

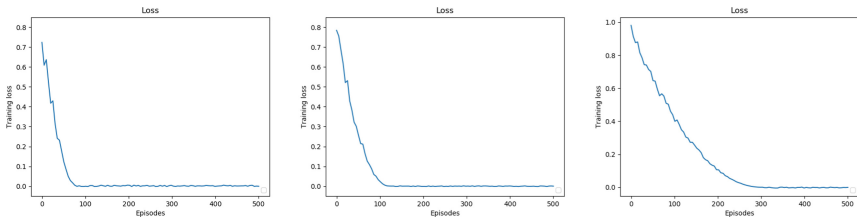
- 1: Randomly initialize the actor-critic network with weights θ_a and θ_c , $\theta_a^{old} \leftarrow \theta_a$
- 2: **for** episode $k = 1, 2, \dots, K$ **do**
- 3: **for** $n = 1, 2, \dots, N$ **do**
- 4: $W_k^n = w_n$
- 5: $V_k^n = v_n$
- 6: **end for**
- 7: Get the set of available workers $W_k = \{W_k^1, \dots, W_k^n\}$ and the set of available tasks $V_k = \{V_k^1, \dots, V_k^n\}$
- 8: Get the current state set $s_k = \{W_k, V_k\}$
- 9: Get the current action a_k according to the policy $\pi(a_k | s_k, \theta_a^{old})$
- 10: Get the reward r_k and the state in the next iteration s_{k+1}
- 11: Calculate the A_k and $Value_k$ in critic network
- 12: Store the set $s_{k+1}, s_k, a_k, r_k, A_k, Value_k$ into D
- 13: **if** $t\%|D| == 0$ **then**
- 14: $\Delta\theta_a = \frac{1}{|D|} \sum_{j=1}^{|D|} \{[r_j + \gamma Value(s_{j+1}; \theta_v) - Value(s_j; \theta_v)]^2, A_k\}$
- 15: Update the θ_a using $\Delta\theta_a$ according to the Gradient Up Strategy
- 16: $\theta_a^{old} \leftarrow \theta_a$, Clear D
- 17: **end if**
- 18: **end for**

4 Experiments and Results

4.1 Experimental Settings

In this paper, Gowalla and TaskMe datasets are selected to conduct simulation experiments, from which the location and time information of some data are captured. We add randomly generated data in a certain range as other information, such as task reward. Finally, a set of simulation data with 2000 tasks and 1000 workers is generated. The rewards for each task are set to a range of 8 to 20 and randomly generated according to the normal distribution of $N \sim (12, 4)$, while the time for each task is set to a range of 10 to 60. Depending on the requirements of the experiment, part of data from this dataset is selected for simulation experiments in a 200×200 square sensing region.

We first set up a comparison experiment of loss under different number of workers and tasks. In a 200×200 square sensing area space, 80 tasks and 5 workers, 300 tasks and 15 workers, 800 tasks and 30 workers are tested respectively. Then we compare our method with the existing traditional methods, greedy algorithm [12] and the ant colony algorithm (ACO) [6], and the DDQN-based algorithm [11] in terms of convergence speed, maximum benefit and task coverage. The number of ants, the number of iterations and the probability of random selection in ACO are set to 10, 30000 and 0.1 respectively. For the DDQN-based algorithm, the replay memory capacity is set to 10000 times and the number of iterations is 30000 times. The probability of random selection is set to start at 0.9 and gradually decay to 0.1. The effectiveness of privacy protection is verified by an ablation experiment. We compare PPO-based algorithm with the algorithm of DDQN and PPO-based algorithm with differential privacy. The experimental settings are the same as the comparison experiments.



(a) 80 tasks & 5 workers (b) 300 tasks & 15 workers (c) 800 tasks & 30 workers

Fig. 4. The loss function for different states of tasks and workers.

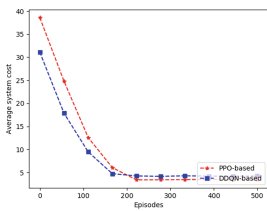


Fig. 5. Average system cost.

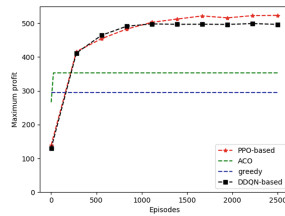


Fig. 6. Maximum profit.

4.2 Loss of Model

Simulation experiments with different numbers of workers and tasks are carried out. In Fig. 4(a), 80 tasks and 5 workers are set. At this time, the number of iteration is within 100 times, and convergence is reached at about the 70th time. Figure 4(b) are set with 300 tasks and 15 workers, and the model converges when the number of iterations is about 120. In Fig. 4(c), 800 tasks and 30 workers are set, and the model converges at 280 iterations. As the number of workers and tasks increases, the convergence rate slows down.

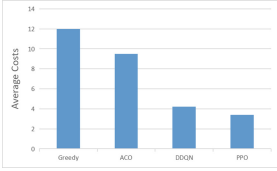


Fig. 7. Average costs.

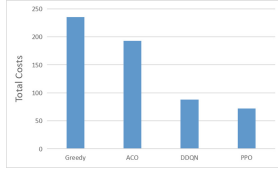


Fig. 8. Total costs.

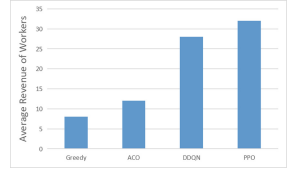


Fig. 9. Average revenue of workers.

4.3 Comparative Experiment

The concept of task coverage is first introduced here: when a task is assigned and completed within its acceptable time range, it can be called the task is covered, so task coverage can be defined as the number of tasks assigned and The ratio of the total number of tasks.

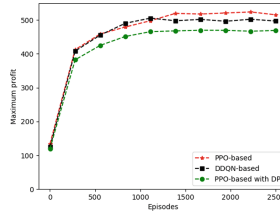
As shown in Table 1, in terms of the maximum profit and task coverage of the platform, the two DRL algorithms based on DDQN and PPO are much higher than traditional methods. And the algorithm based on PPO is better than the algorithm based on DDQN. In Fig. 5, we can see the convergence of the two DRL algorithms based on DDQN and PPO in the average cost of the system. The results show that although the algorithm based on DDQN can achieve faster convergence than our algorithm, our algorithm can achieve smaller average system overhead with 3.4. Similarly, Fig. 6 shows the platform profit of the four algorithms. The greedy algorithm and ACO do not need multiple iterations because they are static algorithms, but their platform benefits are far from the DRL-based algorithm. The DDQN-based algorithm also suffers from faster convergence, but the PPO-based algorithm can ultimately achieve the maximum profit with 518. Figures 7, 8, 9 show the comparison of the four algorithms in terms of average cost, total costs, and average revenue of workers. The results show that the PPO-based algorithm in this paper is superior to other algorithms, while the greedy algorithm performs the worst.

Table 1. Comparison of maximum profit and coverage ratio of four algorithms.

Algorithm	Greedy	ACO	DDQN	PPO
Max profit	298	352	502	512
Coverage ratio	0.4	0.52	0.72	0.76

4.4 Ablation Experiment

As shown in Fig. 10, an ablation experiment for the effectiveness of differential privacy is performed. We compare PPO-based algorithm with DDQN-based algorithm and PPO-based algorithm with differential privacy. The results show that the performance of removing differential privacy can be better. The maximum profit of PPO-based algorithm with differential privacy is lower than other two algorithm due to the blurred information which affects the computational performance of our model. But the method of adding differential privacy can guarantee the privacy protection of workers' information without sacrificing too much performance.

**Fig. 10.** Ablation experiment.

5 Conclusion

In this paper, considering the rationality of task assignment mechanism and the privacy protection of worker information in the dynamic system where the location and status information of workers and tasks are constantly changing, the perceptual task assignment problem of MCS is modeled as an optimization problem for dynamic planning based on discrete datasets. The related algorithms based on differential privacy and DRL are used to solve this problem. In each iteration, we use joint constraints and try to maximize the benefits of the platform while increasing the cumulative benefits of workers as much as possible, so as to continuously optimize the distribution strategy in such a dynamic system. In addition, a differentiated privacy method has been added to the interaction between workers and platform to protect the workers' privacy. The experimental results also demonstrate the validity of this method.

References

1. Mnih, V., et al.: Human-level control through deep reinforcement learning. *Nature* **518**(7540), 529–533 (2015)
2. Cheung, M.H., Southwell, R., Hou, F., Huang, J.: Distributed time-sensitive task selection in mobile crowdsensing. In: *Proceedings of the 16th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pp. 157–166 (2015)
3. Dwork, C.: Differential privacy: a survey of results. In: Agrawal, M., Du, D., Duan, Z., Li, A. (eds.) *TAMC 2008*. LNCS, vol. 4978, pp. 1–19. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-79228-4_1
4. Ganti, R.K., Fan, Y., Lei, H.: Mobile crowdsensing: current state and future challenges. *IEEE Commun. Mag.* **49**(11), 32–39 (2011)
5. Golden, B.L., Levy, L., Vohra, R.: The orienteering problem. *Nav. Res. Logist. (NRL)* **34**(3), 307–318 (1987)
6. Li, W., Jia, B., Xu, H., Zong, Z., Watanabe, T.: A multi-task scheduling mechanism based on ACO for maximizing workers benefits in mobile crowdsensing service markets with the internet of things. *IEEE Access* **7**, 41463–41469 (2019)
7. Liu, Q., et al.: A survey on deep reinforcement learning. *Chin. J. Comput.* **41**(1), 1–27 (2018)
8. Mnih, V., et al.: Playing atari with deep reinforcement learning. *Computer Science* (2013)
9. Mnih, V., et al.: Asynchronous methods for deep reinforcement learning. In: *International Conference on Machine Learning*, pp. 1928–1937. PMLR (2016)
10. Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., Riedmiller, M.: Deterministic policy gradient algorithms. In: *International Conference on Machine Learning*, pp. 387–395. PMLR (2014)
11. Tao, X., Song, W.: Task allocation for mobile crowdsensing with deep reinforcement learning. In: *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–7. IEEE (2020)
12. Weerdts, M., Zhang, Y., Klos, T.: Multiagent task allocation in social networks. *Auton. Agent. Multi-Agent Syst.* **25**(1), 46–86 (2012)



Information Sources Identification in Social Networks Using Deep Convolutional Neural Network

Jiale Wang^(✉), Jiahui Ye, Wenjie Mou, Ruihao Li, and Guangliao Xu

School of Computer Science and Technology, Northwest University, Shaanxi, China
2017117396@stumail.nwu.edu.cn

Abstract. With the ubiquity of electronic communication devices, detecting the information sources is a critical task in reducing the damage caused by malicious sources. However, in the contemporary research of sources identifications and information propagation identifications are calculated through social network topology structure or mathematics inference. In this paper, we borrow the training tool of neural network and propose a deep convolutional neural network to identify the sources in social networks. Initially, we utilize the 20% of data set to play the role of training set and substitute into the proposed model. Subsequently, we employ a bi-graph to classify the trained sources into truth or rumor vertexes. Finally, we utilize our proposed model to test 80% of data set as evaluation results of our identification mechanism. From the experimental results, our developed method can identify more than 85% of information sources and the classification accuracy can reach 80% in both test and train process. The obtained results further indicate that our model can effectively and accurately identify the information sources with reasonable computation costs.

Keywords: Social network · Sources identification · Deep convolutional neural network · Bi-graph

1 Introduction

At present, information spreading in social network have been a common phenomenon to our world. However, unauthorized or malicious people may spread some fake information to cause uncountable damage such as “Obama was injured in two explosions of White House” causes 10 billion U.S. dollars losses within a few hours. This indicates that an insignificant rumor can lead great losses to our individuals and society [1]. With the advance of wireless communications and mobile devices, multiple users can share any information or personal ideals by assessing online social networks [2].

For the purpose of preventing fake information release and spread in social networks, it is significantly essential to identify information sources in complex social networks. However, mostly existing research is concentrated on relationships between individuals, which can identify the information sources by tracing the root of obtained spanning tree based on these relationships [3–6]. Subsequently, researchers utilize vertex centrality to

detect information sources due to sources tend to have higher degrees in social networks [7]. Nevertheless, all of these methods are based on the graph structure properties or graph topology mathematical statistics principles [8–12]. With the expansion of social networks, the cost of computation is extremely increased for traditional identification techniques.

To address this problem, we propose a novel information identification method by utilizing a deep convolutional neural network. We will focus on following challenges. Initially, to transfer a high-level network graph data into a low-level vectors structure, we utilize graph embedding [13] to extract features. To identify the information sources in complex communication networks, we develop a deep neural network to train the test embedding data. Finally, to determine accurately the sources from trained suspicious vertexes, we utilize credibility factors of each suspicious vertexes and evaluate the final identification results [14].

Our main contributions can be summarized as following illustrations:

- To best of our knowledge, we are the first to develop a deep neural network for addressing the information sources identification with reasonable computation costs. Comparing with existing identification models [15–17], our proposed mechanism can achieve much more higher identification accuracy. Extensively experimental results indicate that our designed mechanism can obtain the sources with reasonable computation costs.
- We have considered graph embedding to dispose graph structure dataset, which can be used in subsequent neural network that requires Euclidean space format. Moreover, we develop extractor to obtain the social network features including graph properties and relationship among each vertexes.
- We propose to utilize a bi-graph to distinguish the high credibility vertexes and opposite low credibility vertexes in social networks, which is used to identify the information sources after deep neural network training process.

2 Related Works

In this section, we illustrate recently research outcomes and methods that is related to address the information sources identification in social networks.

As we discussed in Sect. 1, identification methods have been proposed to detect information sources in term of time-stable and time-varying social networks. Most recent research is concentrated on time-varying and utilize inference strategy to detect information sources. Based on this background, a joint inference is applied to identify sources [18], which aims to calculated the credibility for each users in social network and query each users to obtain the final sources. Apparently, querying for each vertexes will cost enormous computation resources and the accuracy is likely interference by inference process.

Currently, researchers concentrate on time-varying social network with time-varying topology structures and users, which is more satisfies the real social networks. In practice, a identification mechanism for time-varying social network is designed [19], which is utilizing a series of time-integrating windows to transfer the dynamic network topology

structure into a stable format and a maximum likelihood function is applied to enhance the accuracy of sources identification.

To address these issues, we develop a deep neural convolutional network to train and test the information sources in social networks and further demonstrate the model evaluation and analysis association of embedding graph in designed neural network.

3 Notations and Preliminaries

3.1 Notations

We define the network as $G = \{V, A, P\}$, where V represents the users set in social network, A represents the adjacency matrix and P is the attributes matrix of social network. Subsequently, we utilize the symbol u to indicate a single user in social network where $u \in V$ and the single vertex u embedding result represents Eu and the whole network embedding is represented as EG . We summarize the primary used notations and corresponding description as following Table 1 demonstration.

Table 1. Summary of primary notations used in this paper.

Notation	Description
$G = \{V, A, P\}$	Nodes set V , adjacency matrix A and attributes matrix P
$f(u, v)$	Extracted features among vertex u and v
δ	Credibility of vertexes
T	Time slot of time-varying network

3.2 Deep Convolutional Neural Network

When applying a deep convolutional neural network, input layer begins to obtain the features of data. After detail functional layers, the classification results is produced by developed layers. In this paper, we primary use convolutional layer that is consisted by convolutional kernels to identify input-data features and produce the convolutional calculation to subsequent layer, pooling layer scales down the features of upper layer, fully connected layer is transferring a sequence array as the output and final softmax layer will output 10 probability distribution of classification results.

3.3 Information Sources Identification

Information Sources Identification in Social Network. In traditional social networks, each user is simplified as a vertex and the relationship is represented by an edge in simulated network. Information can spread among these edges in simulated network and sources identification is concentrated to search the first vertex of telling the message to others among these spread routes.

4 System Model

The whole system framework is depicted in Table 1. The designed method is consisted by three components including graph embedding, construction of deep convolutional neural network and bi-graph to identify spreading sources. We adopt graph embedding tool to transfer network graph state into low dimensional vectors, which is appropriated as training data in neural network system. The deep convolutional neural network is the core of whole designed model, which utilized the embedding graph data to classify the suspicious information sources through its spreading paths and evaluate the credibility of vertexes by graph properties. A bi-graph is applied in the final of designed model to classify the different credibility vertexes and identify the information spreading sources from the classification results (Fig. 1).

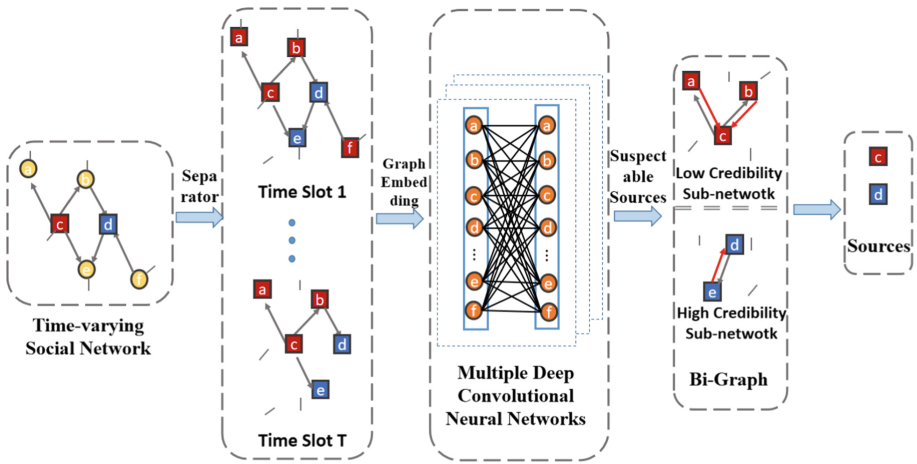


Fig. 1. The system framework of proposed model.

4.1 Graph Embedding

After transferring the social network into time-varying graphs, the DCNN model is hard to dispose the networks including nodes and edges. Therefore, graph embedding is applied to solve this challenge. In this paper, we utilized Embedding Projector service, which is provided by Google to visually generate the embedding properties.

4.2 Deep Conbolutional Neural Network

General structure is achieved by multiple T DCNNs with same inner kernels and layers corresponding T time slots. For simplification, we demonstrate one inner DCNN structure as the following Table 1 demonstration. Convolutional contain the features of each vertex represent as $f(u)$ and calculate convolution process for each embedding vectors, which is represented by symbol $f(u,v)$. The pooling layer and rectified linear layer is

utilized to eliminate the redundancy edges in information spreading process. Two full connected layers investigate all possible sources after information spreading process. The whole iterations is represented in Eq. 1.

$$C_v^{l+1} = U_{l+1}(C_v, \sum_{u \in \text{neigh}[v]} M_{l+1}(O_l^{f(v)}, P_l^{f(u,v)}, R_l^{f(u,v)}, F_l^{512}, F_l^8)) \quad (1)$$

where the C_v^{l+1} is the DNCC $l + 1$ iteration result for vertex v , U_{l+1} is the DNCC operation procedure. The C_v is current embedding vector state and M_{l+1} function is the corresponding layers in proposed structure. Therefore, the credibility of vertex δ_{ij} toward information I_j can be expressed as:

$$\delta_{ij} = \frac{\delta_i^1 + \delta_i^{-1}}{2} \quad (2)$$

where $\delta_i^1 = P(I_j=1|v_i = 1)$, which means the probability that user v believes that information I_j is a real information and information I_j itself is a real information. Otherwise, $\delta_i^{-1} = P(I_j=0|v_i = 0)$, representing the probability that user v believes that information I_j is fake information and information I_j is really fake and unreliable.

Initially, the information randomly contains a certain reliable value represent as $I_j \in (0, 1)$, where 1 means an extreme true information. The covered credibility will replace the previous values and the following equation describes coverage procedure (Fig. 2).

$$\delta_i = \frac{1}{1 + \left(\frac{\alpha_i}{1-\alpha_i}\right)^V \prod_{i \in V} \left(\frac{1-\delta_i^{1(0)}}{\delta_i^{1(0)}}\right)^{\frac{1+v_i}{2}} \left(\frac{\delta_i^{0(0)}}{1-\delta_i^{0(0)}}\right)^{\frac{1-v_i}{2}}} \quad (3)$$

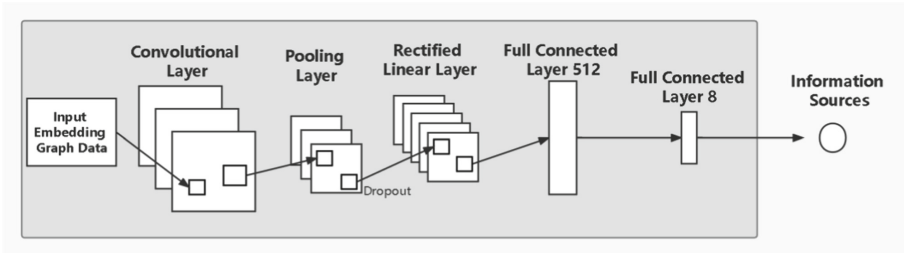


Fig. 2. Deep convolutional neural network structure.

4.3 Sources Identification

Algorithm 1: Information Sources Identification

Input: Suspicious vertexes set S , vertex credibility δ_i

Output: Information sources set R

- 1 for each information j do
 - 2 Dividing suspicious vertexes into two sub-networks(Bi-graph) .
 - 3 for each sub-network do
 - 4 Searching previous vertex in spreading path. Construct filter eliminates sequences.
 - 5 if previous vertex $U_{\text{filter}} = \emptyset$ do
 - 6 Current vertex $\in R$
 - 7 Return Information sources set R
-

Our solution is to establish two sub-network, one contains high credibility users and another contains low credibility users. Subsequently, the appearance sequence of suspicious users and the spreading path are two critical factors to determine sources. Algorithm 1 demonstrates the detail procedure of identifying the information sources.

5 Experiments

5.1 Datasets

We utilize gemsec-Deezer to create a simulation situation and train the DCNN, and Emergent works as the testing situation for proposed model. The detail description about these datasets is demonstrating as follows:

Gemsec-Deezer: The dataset is consisted by 143844 vertexes and 846915 edges from European countries, which is open access and available in SNAP website. We randomly generate 200 real information and 200 fake information to spread in this situation by utilizing the proposed mechanism to identify the sources.

Emergent: Dataset is concluded by a digital journalism process and obtains totally 102 true information spreading among the web and 78 fake information.

5.2 Experiment Setup

We simulate our propose model in above described datasets with same machine. The evaluation contains accuracy, distribution and cost aspects. The Identification accuracy is calculated the percentage of detected sources and original information spreading sources. The distribution of credibility is obtained the evaluation 10-levels of estimated users credibility and real situation credibility in datasets. The computation cost measures the performance of propose model.

Additionally, we compare our proposed mechanism with existing information sources identification algorithms, which is introducing as follows:

- Source-CR detects sources by inferring and querying each vertex.
- Rumor Source Identification is an algorithm to search the spreading sources in time-varying topology structure [19].
- MVNA algorithm is a method that infers the information sources by users properties in social network.

6 Model Evaluations

6.1 Performance Evaluation

We simulate the information spreading in mentioned datasets and utilize the identification accuracy to evaluate proposed model, which is plotted in Table 1. We can significantly observe that the detection accuracy is continuously increasing with the expansion of information pieces. The available extracted features appears more frequently than less pieces condition that means the DCNN can obtain higher accuracy for identifying the suspicious sources. When the piece of information reaches around 600, the trained accuracy is closely reached at approximately 80%, which is an acceptable accuracy for sources identification (Fig. 3).

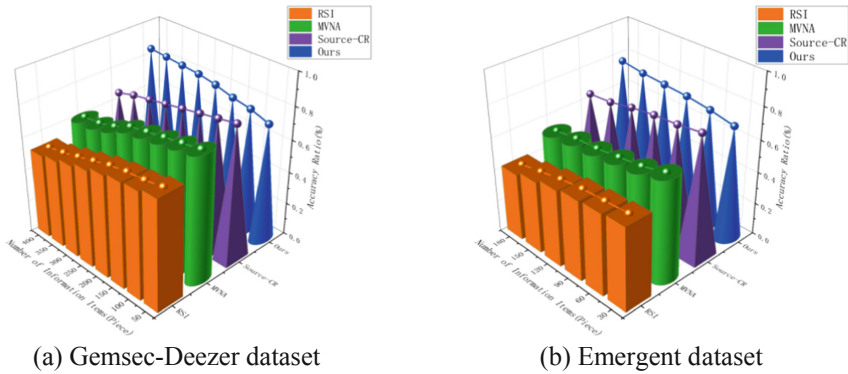
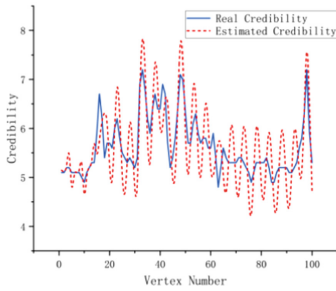


Fig. 3. Model performance of identification accuracy and comparison results in (a)Gemsec-Deezer and (b)Emergent dataset respectively.

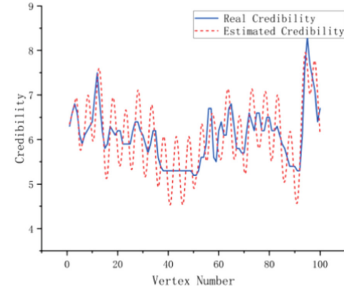
Figure 1 respectively demonstrates the credibility evaluation distribution of proposed mechanism in mentioned dataset. We select ahead 100 vertexes in datasets to observe the distribution situation and utilize 10-levels to measures the credibility of vertex, which means $\delta_{ij} \in [1, 10]$. We can significantly observe that our estimated values is well matched in original vertex credibility and amplify the real credibility, which is convenient for the process of classification in bi-graph (Fig. 4, Table 2).

6.2 Cost Evaluation

We simulate the existing sources identification algorithms and our method in same experimental setting conditions. Table 1 demonstrates computation costs when identification algorithm faces 100 information spreading.



(a) Gemsec-Deezer dataset



(b) Emergent dataset

Fig. 4. Model performance of credibility distribution results in (a)Gemsec-Deezer and (b)Emergent dataset respectively.

Table 2. Computation cost comparison result.

Dataset	Gemsec-Deezer	Emergent
RSI Computation cost (second)	102	87
MVNA Computation cost (second)	95	75
Source-CR Computation cost (second)	124	95
Ours Computation cost (second)	118	93

7 Conclusion and Future Improvements

In this article, we propose a novel information sources identification mechanism by utilizing credibility to identify suspicious sources and determining the sources by constructing a bi-graph. We evaluate identification accuracy and assess our method with reasonable cost. As for future improvement, we will search other substitute credibility factors to decrease the computation speed and costs.

References

1. Starbird, K., Maddock, J., Orand, M., Achterman, P., Mason, R.M.: Rumors, false flags, and digital vigilantes: misinformation on twitter after the 2013 boston marathon bombing. In: Proceedings iSchools IConference, pp. 654662 (2014)
2. Shu, K., Sliva, A., Wang, S., Tang, J., Liu, H.: Fake news detection on social media: A data mining perspective. In: Proceedings ACM SIGKDD, pp. 2236 (2017)
3. Murnane, K., Bayen, U.J.: An evaluation of empirical measures of source identification. *Mem Cogn* **24**, 417–428 (1996)
4. Feng, X., Zhang, W., Zhang, Y., Xiong, X.: Information identification in different networks with heterogeneous information sources. *J. Syst. Sci. Complexity* **27**(1), 92–116 (2014)
5. Shu, K., Bernard, H.R., Liu, H.: Studying fake news via network analysis: detection and mitigation. *Emerging Research Challenges and Opportunities in Computational Social Network Analysis and Mining*, pp. 4365 (2019)

6. Choi, J., Yi, Y.: Necessary and sufficient budgets in information source finding with querying: Adaptivity gap. In: Proceedings IEEE ISIT, pp. 22612265 (2018)
7. Choi, J., Shin, J., Yi, Y.: Information source localization with protector diffusion in networks. *J. of Commun. Networks* **21**(2), 136147 (2019)
8. Yang, F., Liu, Y., Yu, X., Yang, M.: Automatic detection of rumor on sina weibo. In: Proceedings ACM SIGKDD, pp. 1319 (2012)
9. Ruchansky, N., Seo, S., Liu, Y.: Csi: a hybrid deep model for fake news detection. In: Proceedings ACM CIKM, pp. 797806 (2017)
10. Shah, D., Zaman, T.: Rumors in a network: Who's the culprit?. *IEEE Trans. Information Theory* **57**, pp. 51635181 (2011)
11. Jiang, J., Wen, S., Yu, S., Xiang, Y., Zhou, W.: K-center: An approach on the multi-source identification of information diffusion. *IEEE Trans. Information Forensics and Security* **10**, pp. 26162626 (2015)
12. Chen, Z., Zhu, K., Ying, L.: Detecting multiple information sources in networks under the SIR model. In: 2014 48th Annual Conference on Information Sciences and Systems. IEEE, pp. 14 (2014)
13. Basaras, P., Katsaros, D., Tassioulas, L.: Detecting influential spreaders in complex, dynamic networks. *Computer* **4**, pp. 2429 (2013)
14. Luo, W., Tay, W.P., Leng, M.: Rumor spreading and source identification: a hide and seek game. arXiv preprint [arXiv:1504.04796](https://arxiv.org/abs/1504.04796) (2015)
15. Wang, Z., Dong, W., Zhang, W., Tan, C.W.: Rumor source detection with multiple observations: Fundamental limits and algorithms. In: The 2014 ACM International Conference on Measurement and Modeling of Computer Systems, Series. SIGMETRICS' 14. ACM, pp. 113 (2014)
16. Zhu, K., Ying, L.: Information source detection in the SIR model: a sample path based approach. In: Information Theory and Applications Workshop (ITA), pp. 19 (2013)
17. Wang, D., Kaplan, L., Le, H., Abdelzaher, T.: On truth discovery in social sensing: a maximum likelihood estimation approach. In: Proceedings ACM IPSN, pp. 233244 (2012)
18. Qu, S., Zhao, Z., Fu, L., Wang, X., Xu, J.: Joint inference on truth/rumor and their sources in social networks. *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*. IEEE Press, pp. 924933 (2020)
19. Jiang, J., Wen, S., Yu, S., Xiang, Y., Zhou, W.: Rumor source identification in social networks with time-varying topology. *IEEE Trans. Dependable and Secure Computing* **15**(1), 166–179 (2018)

Underwater and Underground Networks



MineTag: Exploring Low-Cost Battery-Free Localization Optical Tag for Mine Rescue Robot

Xiaojie Yu¹, Xu Yang^{1,3}(✉), Yuqing Yin¹(✉), Shouwan Gao¹, Pengpeng Chen^{1,2}, and Qiang Niu^{1,2}

¹ School of Computer Science and Technology, China University of Mining and Technology, Xuzhou 221116, China

{yuxiaojie,yang_xu,yinyuqing,gaoshouwan,chenp,niuq}@cumt.edu.cn

² Mine Digitization Engineering Research Center of Ministry of Education of the People's Republic of China, China University of Mining and Technology, Xuzhou 221116, China

³ Xuzhou Kerui Mining Technology Co., Ltd., Xuzhou 221116, China

Abstract. Inertial navigation adopts localization base stations to correct cumulative errors for mine rescue robots, while requirements of explosion-proof safety hinder the application of regular powered base stations in harsh coal mine environments. Therefore, we propose MineTag, a novel localization base station for self-positioning of coal mine robots, which is built with low-cost and battery-free optical tags via a differ-neighbor deployment strategy. The main innovation of the tag is to modulate the light retro-reflection with a light absorption mechanism, allowing the tag to reflect a specific light intensity without the need for a power source. According to the topological relationship of tags, we propose a novel tag recognition algorithm based on trajectory matching to determine which tag the robot is under. Finally, we implemented MineTag and evaluated its performance in a real coal mine. Experimental results show that MineTag can achieve the tag recognition accuracy of more than 95%, and the localization accuracy is 98% error of 2.6 m or less.

Keywords: Underground self-positioning · Optical tag · Battery-free

1 Introduction

Complex environment in coal mine tends to cause disasters [1, 2], accompanied by power outages, equipment damage, gas leakage, and other dangerous situations. From the year 2004 to 2021, there were 21753 coal mine disasters in China. In

This work was supported by the National Natural Science Foundation of China under Grant 51904294 and the Natural Science Foundation of Jiangsu Province under Grant BK20221109.

2021, a coal and gas outburst disaster occurred in Coal Mine in Heilongjiang Province. In coal mine rescue, the time to determine the location of missing miners accounted for about half of the total rescue time (about 16 h), while the rescue robot played a great significant role. The self-positioning of the robot is the basis for any rescue task [3–5].

For long and narrow coal mine environments, the inertial navigation method commonly used for robot localization has a sizeable cumulative error [6, 7]. Towards this end, utilizing the localization base station to correct the error is an ideal method [8]. Despite a wide spectrum of wireless base station (such as WiFi [9–11], Bluetooth [12–14], UWB [15–17] and Zigbee [18, 19]). However, there has been very limited adoption of coal mine. Due to the interference of the shaft wall, wireless signal is severely attenuated [20]. Besides, the mine tunnel tends to fill with gas after the mine accident, while the electromagnetic induction generated by wireless communication increases the possibility of gas explosion. What is more, wireless base station calls for continuous power supply from the power infrastructure. On the one hand, it involves great potential safety hazards and energy consumption. On the other hand, mine accidents are often accompanied by power interruptions, resulting in base station being inoperable. Therefore, building the battery-free base station without safety hazards is the key to realizing the localization of the coal mine rescue robot. RFID-based system [21, 22] as its battery-free is a promising method, which uses backscatter communication technology. Nevertheless, this method also has serious signal attenuation and high cost, hindering the large-scale application in coal mine.

To address this issue, this paper proposes a battery-free localization scheme for coal mine robot, called MineTag. The core of MineTag is to employ the low-cost and battery-free optical tags to build localization base station. Then the mine robot obtains the location information of the location-marked tags for self-positioning. Minetag mainly includes the following steps. First, to adapt to the complex coal mine environment with high safety requirements, we designed a low-cost and battery-free optical tag based on the idea of modulating the light retroreflection with light absorption mechanism, which requires no power supply and no electromagnetic interference. Second, to enable large-scale underground applications with limited optical tags, we propose the differ-neighbor tag deployment strategy to build localization base station with reused optical tags. Finally, considering that the light intensity is no longer the unique identifier of the tag due to the tag reuse, we introduce the topological relationship to propose a novel tag recognition algorithm based on trajectory matching. The main contributions of this work are summarized as follows:

- To the best of our knowledge, this is the first study to investigate the use of battery-free optical tags to enable mine rescue robot self-positioning, which can be appropriate for harsh mine post-disaster environments.
- We design a novel low-cost and battery-free optical tag based on absorption mechanism. Furthermore, we propose the differ-neighbor tag deployment strategy and trajectory matching-based tag recognition mechanism to enable large-scale application in coal mine.

- We implement MineTag and conduct plenty of real scene experiments and simulations to evaluate its performance extensively. Experimental result shows that MineTag can achieve the tag recognition accuracy of more than 95%, and the localization accuracy is 98% error of 2.6 m or less.

2 System Overview

In this section, we outline the basic design of MineTag. The overall system consists of localization base station composed of optical tags, an LED transmitter and a light sensor. As shown in Fig. 1, we designed a battery-free optical tag, which consists of two parts, an absorption layer and a reflective layer, used to modulate the light intensity and backscatter the incident light respectively. Then, tags with sufficient light intensity resolution are selected as anchor tags to build localization base station with the strategy of differ-neighbor and store the location information of the tags. Furthermore, we designed a series of tag recognition algorithms. First, search for candidate tags based on the current light intensity received by the robot, and then combine the historical light intensity information to extract the historical tag sequence. Finally, introduce the topological relationship of the tags to determine which tag the robot is under by trajectory matching.

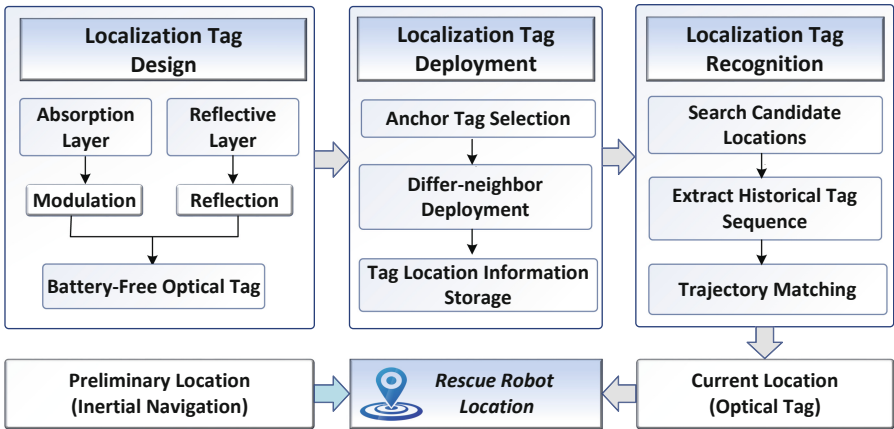


Fig. 1. System overview

3 System Design

3.1 Optical Tag Design

The core of MineTag is the low-cost and battery-free optical tag, which adopts a novel model based on light absorption mechanism to modulate the intensity

of reflected light. Before describing the tag's design, it is helpful to understand the design principles of the tag.

Principle. An essential function of the tag is to modulate the optical signal. In the previous work, in order to realize the modulation function, the tag needs to be equipped with power supply and chip, resulting in high cost and high power consumption of the system. For the sake of battery-free and chip-less, this system envisages changing the optical properties of the tag itself. As light passes through a medium, the particles in it will absorb part of the light and weaken its intensity. It is worth noting that the light absorption capacity of the medium is affected by its material properties, such as color and thickness. We utilize this feature to modify the tag. The film structure is used to construct the tag, then the light absorption capacity of the tag can be controlled by changing the color and thickness of the film. In the following, we discuss the influence of film thickness and color on light absorption performance.

(a) Influence of film thickness. We introduce a typical absorption model, Lambert-Beer law, to illustrate the modulation ability of film thickness [23]. Lambert-Beer law describes the absorbance of the medium, the thickness of the medium, and the concentration of the light-absorbing substance. The mathematical expression is as follows:

$$L = \log(I_0/I) = a \times b \times c \quad (1)$$

where L is the absorbance of the medium, I_0 is the intensity of incident light, I is the intensity of transmitted light. a is the wavelength-dependent absorptivity coefficient, which is a qualitative value, b is the path length of the light through the medium, c is the medium concentration. When the temperature and other conditions remain constant, the absorbance of the medium L is proportional to the medium concentration c and the path length of the light b , while the path length is closely related to the thickness of the medium. When increase the number of layers of the film, the concentration of the medium does not change, with the thickness increasing, resulting in an increase in the absorbance of the medium. Thus, L is positively correlated with the thickness of the film, which allows the tag to modulate the light intensity with different film layers.

(b) Influence of film color. The modulation effect of film thickness on light intensity is limited, so we need to find more modulation methods. Selective absorption of light by medium is an ideal physical phenomenon. The electronic structure of medium is different, and the wavelength of light that can be absorbed is also different, which constitutes the basis of selective absorption of matter [24].

$$\lambda = \frac{hc}{\delta} \quad (2)$$

where λ is the wavelength of absorption, δ is the valence electron transition energy difference, which is related to the electronic structure of matter, h is the planck constant, c is the light speed. The light source is a polychromatic light composed of light of different wavelengths. When the color of the film is changed,

the resulting electronic structure change will lead to the film selectively absorbing light of different wavelengths. It means films with different colors have different light absorption capabilities, leading to differences in intensity modulation.

Tag Structure. As shown in Fig. 2, the tag is composed of two parts: an absorption layer and a reflective layer. The reflective layer on the bottom reflects lights back to the light source, the reflected lights then pass through the absorption layer with different layers and colors, which has different light absorption capabilities, resulting in a specific light intensity for light passing through the tag. Thus, the light intensity modulation is realized by changing the optical properties of the tag itself to battery-free.

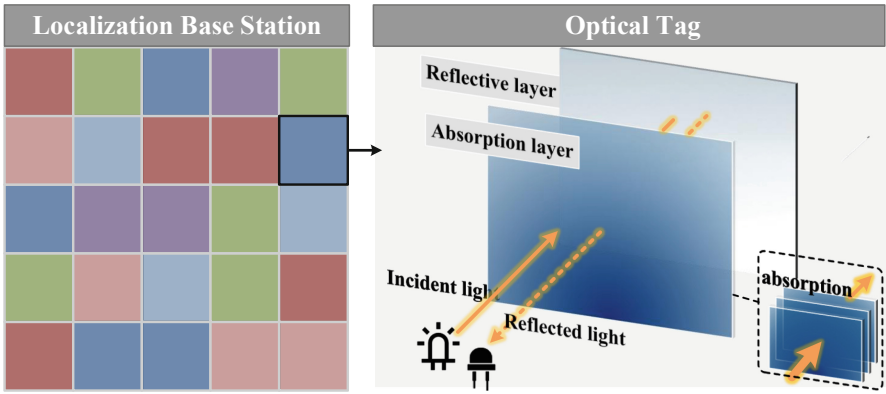


Fig. 2. Tag structure

3.2 Build Localization Base Station

Anchor Tag Selection. It is noted that optical tags must have sufficient light intensity resolution to avoid being difficult to distinguish each other. Therefore, we set the threshold s to filter the tags. Only when the difference in light intensity modulated by the two tags is greater than the s , the tags are selected as the anchor tags to build localization base station.

Tag Deployment. Since there are only a limited number of anchor tags, positioning with a single tag cannot satisfy large-scale deployment in coal mine. To solve this problem, we unite the tags to build the localization base station. As shown in Fig. 3, divided the localization base station into $p \times q$ grids, suppose there are n anchor tags, $n < (p \times q)$, each anchor tag will be used multiple times to fill the base station. In order to avoid path crossing during subsequent trajectory matching, we design a differ-neighbor deployment strategy, that is, the tags of adjacent positions of a tag cannot be the same. If the grid position of a

tag is $G(X_i, Y_j)$, the tags at its neighbor $G(X_{i+1}, Y_j)$, $G(X_{i-1}, Y_j)$, $G(X_i, Y_{j+1})$ and $G(X_i, Y_{j-1})$ should be different from each other. Furthermore, store the tag location information in each base station.

Optical Tag Deployment				
$G(X_i, Y_i)$ Tag _(i,i)	...	$G(X_i, Y_{j-1})$ Tag _(i,j-1)	$G(X_i, Y_j)$ Tag _(i,j)	$G(X_i, Y_{j+1})$ Tag _(i,j+1)
...
$G(X_{i-1}, Y_i)$ Tag _(i-1,i)	$G(X_{i-1}, Y_j)$ Tag _a	...
$G(X_i, Y_i)$ Tag _(i,i)	...	$G(X_i, Y_{j-1})$ Tag _b	$G(X_i, Y_j)$ Tag _(i,j)	$G(X_i, Y_{j+1})$ Tag _d
$G(X_{i+1}, Y_i)$ Tag _(i+1,i)	$G(X_{i+1}, Y_j)$ Tag _c	...

Fig. 3. Optical tag deployment

3.3 Optical Tag Recognition

In this section, we introduce the algorithm for tag recognition. First, search for candidate locations based on the current light intensity received by the robot, and then combine the historical light intensity information to extract the historical tag sequence. Finally, introduce the topological relationship of the tags to identify which tag the robot is under based on the trajectory matching.

Search Candidate Locations. The robot moves in the area where the localization base station is laid, the receiver obtains the reflected light intensity modulated by the tag in real-time. First, determine which anchor tag the robot is under according to the received light intensity. Assume that the mathematical set of all anchor tags is as follows:

$$T = tag_1, tag_2, \dots, tag_m \tag{3}$$

where m is the label number of the anchor tag. The fingerprint of anchor tags and received light intensity has been established, The mathematical set of fingerprints corresponding to T is shown as follows:

$$I = i_1, i_2, \dots, i_m \tag{4}$$

where i_m denotes the intensity of the reflected light modulated by the anchor tag m . The reflected light intensity received at the current time is expressed as $i_{current}$. Since the LED voltage fluctuates slightly, when perform fingerprint

database matching, calculate the Euclidean distance between the measured value and the standard value. The anchor tag corresponding to the light intensity value with the minimum Euclidean distance is selected for matching. The calculation formula of the Euclidean distance D is shown as follows:

$$D = |i_{current} - i_m| \quad (5)$$

Due to the sufficient light intensity resolution of anchor tags, the correct matching of tags can be achieved even if the light intensity fluctuation caused by voltage fluctuation. Then, use the label number of the matched anchor tag as the index to find its position. Due to the reuse of tags, multiple positions will be found as candidate positions, one of which is the position of the robot.

Trajectory Matching. Extract the historical intensity information for a while time and convert it into a historical tag sequence $(tag_{current}, tag_{1-step}, tag_{2-step}, \dots, tag_{n-step})$, which can be regarded as the robot's trajectory. We propose a topological relationship-based trajectory matching algorithm to further determine which tag the robot is under. The adjacent tags of the historical tag sequence have an adjacent topology relationship in the localization base station. Search the same path in the localization base station as the historical tag sequence, the starting tag of the path is the tag where the robot is currently under. The trajectory matching process is as follows: take a candidate position as the starting point, take the topological relationship as the constraint, and perform path matching with the localization base station based on the historical tag sequence. If a certain tag is not matched, the path is lost. When there is only one path left, perform another two steps matching on this path to verify the correctness of the path. If the matching is still successful, the path is the true motion trajectory of the robot.

4 Experiment and Evaluation

4.1 Experimental Results

Experimental Setup. We first evaluated the MineTag's performance by conducting small-scale experiments in real scenarios. As shown in Fig. 4, we conducted preliminary tests in test scenarios and deployed in real coal mine ($100\text{ m} \times 3\text{ m} \times 3\text{ m}$). The length of the mine tunnel is about 500 m. The tag size is $40 \times 40\text{ cm}^2$. The reflective layer of the optical tag is made of the retroreflective material commonly used in traffic signs, and the absorption layer is made of the cheaper polyethylene material. The cost of each tag is about 50 cents and even less when in mass production. A rescue robot was used as the carrier. A modulated LED light with 2500 Hz frequency and a PD were mounted on the top of the robot and faced upward. Furthermore, the motion capture system tracks the robot's actual position information in real-time. This experiment uses an

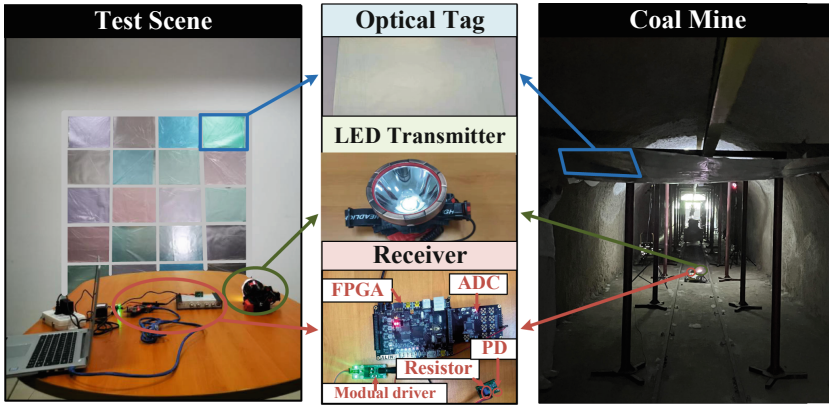


Fig. 4. Experimental scene

off-line localization scheme. First, data are collected in coal mine environment, and then these data are analyzed in the laboratory. We conducted a thousand experiments under each experimental condition.

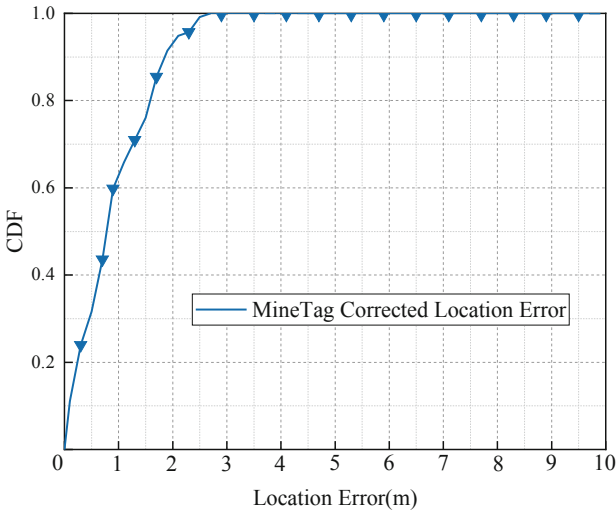


Fig. 5. MineTag corrected localization performance

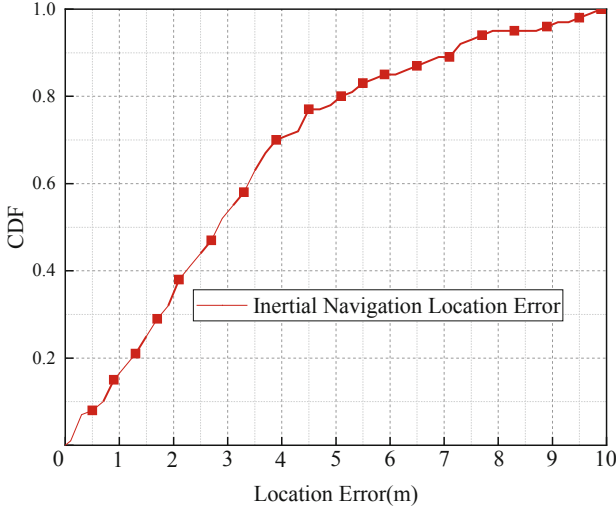


Fig. 6. Inertial navigation localization performance

Localization Performance. In this section, we evaluate the localization performance of the MineTag. The localization base station was deployed every 100 m, Each base station contains 4×7 grids formed by 6 anchor tags. Robot advances in the mine tunnel, Fig. 5 and Fig. 6 show the cumulative distributions(CDFs) of the location error of inertial navigation and MineTag corrected, respectively. It can be seen that the 80% of the location error of inertial navigation is 5.2 m and the maximum location error is up to 9.7 m, while the localization accuracy of MineTag corrected is 98% error of 2.6 m or less.

4.2 Impact of Varying Factors

In MineTag, the localization accuracy largely depends on the recognition accuracy of the optical tag. Next, we evaluate the performance of MineTag in terms of optical tag recognition accuracy under varying factors. We construct $40 \times 40 \text{ cm}^2$, $60 \times 60 \text{ cm}^2$, and $80 \times 80 \text{ cm}^2$ optical tags respectively, and conduct 500 experiments on each experimental condition.

Impact of Ambient Light. We evaluate the tag recognition accuracy under different intensities of ambient light. We change the ambient light intensity by turning on and off different numbers of LED lights and choose seven grades of illuminance: 1, 50, 250, 1250, 2500, 3750, and 5000, for which 1 lux means a dark environment. Figure 7 shows that for the different intensities of ambient light, the average recognition accuracy remains around 95%, with no noticeable impact from the variation of ambient light. It is because that the transmitter is modulated at a specific frequency, which makes it possible to extract the light intensity of the unique frequency of the robot without being affected by the ambient light.

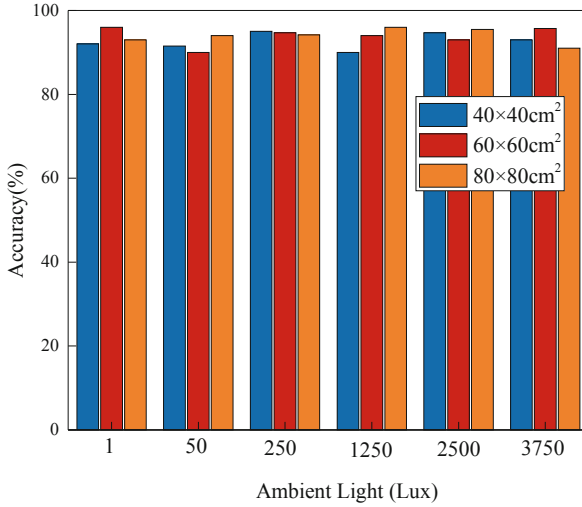


Fig. 7. Impact of ambient light

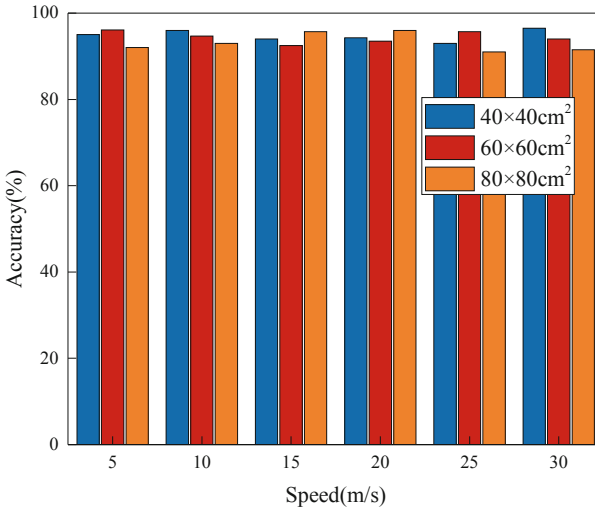


Fig. 8. Impact of speed

Impact of Speed. This section carries out experiments on the influence of robot’s speed on recognition accuracy. We set the speed of the robot to 5 m/s, 10 m/s, 15 m/s, 20 m/s, 25 m/s, Fig. 8 shows the performance of average tag recognition accuracy for the different conditions. The result shows that for all the test speeds, the average recognition accuracy remains around 95%. It implies that speed has no significant impact on accuracy.

5 Conclusion

This paper proposes MineTag, a low-cost battery-free localization scheme for coal mine rescue robot. Our main idea is to employ the low-cost and battery-free optical tags to build localization base station. Then the mine robot obtains the location information of the location-marked tags for self-positioning. Furthermore, we propose the differ-neighbor tag deployment strategy and trajectory matching-base tag recognition mechanism to enable large-scale application in mine. We implemented MineTag in real coal mine and evaluated its performance. Experimental results show that MineTag can achieve the tag recognition accuracy of more than 95%, and the real-time localization accuracy is 98% error of 2.6 m or less. The system has a high potential to provide a low-cost and effective solution for precise positioning of rescue robot in coal mine.

References

1. Xuhui, Z., Runlin, D., Yongwei, L.: VR-based remote control system for rescue detection robot in coal mine. In: 2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI), pp. 863–867 (2017)
2. Li, X., Cao, Z., Xu, Y.: Characteristics and trends of coal mine safety development. *Energy Sources Part A: Recovery Utilization Environ. Effects* 1–19 (2021)
3. Yang, X., Yu, X., Zhang, C., Li, S., Niu, Q.: MineGPS: battery-free localization base station for coal mine environment. *IEEE Commun. Lett.* **25**(8), 2579–2583 (2021)
4. Yang, Y., Li, Y., Guo, X.: Underground personnel positioning system based on low-power card reader. In: International Conference on Automatic Control and Artificial Intelligence (ACAI 2012), pp. 2239–2242 (2012)
5. Wang, Y., Tian, P., Zhou, Y., Chen, Q.: The encountered problems and solutions in the development of coal mine rescue robot. *J. Robot.* **2018** (2018)
6. Cho, S., Moon, S., Seo, W.J.: A dead reckoning localization system for mobile robots using inertial sensors and wheel revolution encoding. *J. Mech. Sci. Technol.* **25**, 2907–2917 (2011)
7. Fauser, T., Bruder, S., El-Osery, A.: A comparison of inertial-based navigation algorithms for a low-cost indoor mobile robot. In: International Conference on Computer Science & Education, pp. 101–106 (2017)
8. Panahandeh, G., Jansson, M.: Vision-aided inertial navigation based on ground plane feature detection. *IEEE/ASME Trans. Mechatron.* **19**, 1206–1215 (2014)
9. Yang, Z., Zhou, Z., Liu, Y.: From RSSI to CSI: indoor localization via channel response. *ACM Comput. Surv. (CSUR)* **46**(2), 1–32 (2013)
10. Chen, Z., Zou, H., Jiang, H.: Fusion of WiFi, smartphone sensors and landmarks using the Kalman filter for indoor localization. *Sensors* **46**(2), 715–732 (2015)
11. Zhang, T., Zhang, K., Liu, D., Chen, P.: CSI-based calibration free localization with rotating antenna for coal mine. In: Liu, Z., Wu, F., Das, S.K. (eds.) WASA 2021. LNCS, vol. 12937, pp. 263–274. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-85928-2_21
12. Bargh, M.S., de Groote, R.: Indoor localization based on response rate of bluetooth inquiries. In: Proceedings of the First ACM International Workshop on Mobile Entity Localization and Tracking in GPS-Less Environments, pp. 49–54 (2008)

13. Giovanelli, D., Farella, E., Fontanelli, D., Macii, D.: Bluetooth-based indoor positioning through ToF and RSSI data fusion. In: 2018 International Conference on Indoor Positioning and Indoor Navigation (IPIN), pp. 1–8. IEEE (2018)
14. Burzacca, P., Mircoli, M., Mitolo, S., Polzonetti, A.: “iBeacon” technology that will make possible internet of things. In: International Conference on Software Intelligence Technologies and Applications & International Conference on Frontiers of Internet of Things 2014, pp. 159–165. IET (2014)
15. Li, M.G., Zhu, H., You, S.Z., Tang, C.Q.: UWB-based localization system aided with inertial sensor for underground coal mine applications. *IEEE Sens. J.* **20**(12), 6652–6669 (2020)
16. Poulouse, A., Han, D.S.: Feature-based deep LSTM network for indoor localization using UWB measurements. In: 2021 International Conference on Artificial Intelligence in Information and Communication (ICAIC), pp. 298–301. IEEE (2021)
17. Seguel, F., Palacios-Jativa, P., Azurdia-Meza, C.A., Krommenacker, N., Charpentier, P., Soto, I.: Underground mine positioning: a review. *IEEE Sens. J.* (2021)
18. Han, Z., Mingxia, C., Shunyan, L.: Research on node location algorithm of Zigbee based on optimized neural network, pp. 693–698 (2020)
19. Zhang, Y., Yuan, G., Lei, C., Lei, X.: Mine wireless localization system based on Zigbee technology. *Zhongzhou Coal* (2010)
20. Szrek, J., Zimroz, R., Wodecki, J., Michalak, A., Góralczyk, M., Worsa-Kozak, M.: Application of the infrared thermography and unmanned ground vehicle for rescue action support in underground mine-the amicos project. *Remote Sens.* **13**(1), 69 (2020)
21. Ge, Z., et al.: Mag-barcode: magnet barcode scanning for indoor pedestrian tracking. In: 2020 IEEE/ACM 28th International Symposium on Quality of Service (IWQoS), pp. 1–10. IEEE (2020)
22. Fang, Y., Cho, Y.K., Zhang, S.: Case study of BIM and cloud-enabled real-time RFID indoor localization for construction management applications (2016)
23. Măntele, W., Deniz, E.: UV-VIS absorption spectroscopy: lambert-beer reloaded. *Spectrochim. Acta Part A: Mol. Biomol. Spectrosc.* **173**, 965–968 (2017)
24. Huang, K., Rhys, A.: Theory of light absorption and non-radiative transitions in F-centres, pp. 74–92 (2000)



TSV-MAC: Time Slot Variable MAC Protocol Based on Deep Reinforcement Learning for UASNs

Yuchen Wu, Yao Liu, Zhao Zhao, Chunfeng Liu^(✉), and Wenyu Qu

Tianjin Key Laboratory of Advanced Networking, College of Intelligence and Computing, Tianjin University, Tianjin, China
cfliu@tju.edu.cn

Abstract. With the increasing variety and number of ocean applications, the underwater transmission of heterogeneous ocean data has become a hot spot in the research field of underwater acoustic sensor networks (UASNs). However, due to lack of flexibility in time slot allocation, the existing multiple access control (MAC) protocols for UASNs cannot be effectively applied to the transmission of heterogeneous ocean data. In order to solve the above problem in UASNs with heterogeneous ocean data, we propose a time slot variable MAC protocol (TSV-MAC) based on deep reinforcement learning. In TSV-MAC, the long short term memory (LSTM) deep learning model is constructed and is trained by considering the usage efficiency of time slots and the data collection condition of underwater nodes. Then, the trained LSTM model is applied to predict the generation and transmission of data from each underwater node and a Q-learning model is adopted to allocate a suitable number of time slots to underwater nodes. The TSV-MAC protocol periodically updates the time slot allocation table, to enable UASNs to adapt the different data packets which are dynamically generated. Finally, the effectiveness of the protocol is verified by extensive simulation results.

Keywords: Underwater acoustic sensor networks · Multiple access control protocol · Deep reinforcement learning · Time slot

1 Introduction

With the development of social economy and progress of science and technology, human activities are gradually expanding from land to sea. Underwater acoustic sensor networks (UASNs) can collect underwater information data for various marine applications continuously and conveniently, so they have received very high attention. At present, the application scenarios of UASNs mainly include underwater environmental and ecological monitoring, marine animal migration monitoring, submarine pipeline monitoring [1, 2]. In the complex and dynamic underwater environment, the data transmission efficiency of UASNs is directly related to the timeliness of data collection, which may largely affect the ability

of ocean applications to match the real-time analysis and scheme adjustment of monitoring data with the real-time underwater environment.

The multiple access control (MAC) protocol plays a very important role in the data transmission efficiency of UASNs. Compared with frequency division multiple access and code division multiple access protocols, time division multiple access (TDMA)-based MAC protocols can effectively avoid the data collision and the information interaction, so as to improve the network transmission efficiency of UASNs [3]. However, it is challenging to design an efficient TDMA-MAC protocol for UASNs because of slow propagation speed of underwater sound, irregular data packet generation and dynamic marine communication environment. The traditional TDMA-MAC protocol [4, 5], which allocate a fixed number of time slots, faces the problems of low time slot utilization and low throughput. To solve such problems, many dynamic TDMA-MAC protocols have been proposed by subsequent researchers [6–13]. These TDMA-MAC protocols dynamically allocate different numbers of time slots to different nodes based on information such as their relative positions, propagation delays or scheduling priorities. However, the existing dynamic TDMA-MAC protocols need a lot of information interaction to ensure the performance of data transmission, which costs a lot of node energy. Due to the limited energy and difficult charging of underwater nodes, it is usually unacceptable for UASNs. In addition, these protocols do not consider the transmission issue of heterogeneous data of ocean applications which is an aspect that can not be ignored in the future UASNs.

In order to further improve the transmission performance and reduce the energy consumption of UASNs, some MAC protocols based on machine learning have been proposed and showed excellent capabilities [14–16]. Nevertheless, there is still a lack of in-depth research on the optimization of dynamic TDMA-MAC protocols based on machine learning in the heterogeneous data transmission scenario of UASNs. Therefore, in this paper, we consider the impact of the large differences in data packet size generated by underwater nodes on time slot utilization. Furthermore, we propose a time slot variable MAC protocol (TSV-MAC) based on deep reinforcement learning for UASNs with heterogeneous ocean data. In TSV-MAC, the deep reinforcement learning module is composed of a long short term memory (LSTM)-deep Q-Learning (DQN) [17]. The LSTM model is used to learn and predict the data generation and transmission situation of each underwater node and apply the prediction results to the dynamic TDMA-MAC protocol, so as to decrease the information interaction between nodes and reduce the energy consumption of UASNs. For further improving the time slot utilization and network throughput of UASNs, the DQN model is adopted to allocate a suitable number of time slots to underwater nodes in advance and periodically update the time slot allocation table to enable UASNs to adapt the different data packets that are dynamically generated. To the best of our knowledge, deep reinforcement learning techniques have not been applied to the MAC protocol for dynamic allocation of the number of time slots in related works. The main contributions of this paper are as follows:

- We propose a time slot variable MAC protocol (TSV-MAC) based on LSTM-DQN for UASNs with heterogeneous ocean data.
- We build a LSTM model to predict the data generation and transmission of each underwater node in order to decrease the information interaction between nodes and reduce the energy consumption of UASNs.
- We propose a time slot allocation strategy based on DQN to improve the time slot utilization and network throughput of UASNs.

The remainder of this paper is organized as follows: Sect. 2 gives the network model and problem description. In Sect. 3, the TSV-MAC protocol is described. Section 4 performs the simulation results. The conclusions is given in Sect. 5.

2 Network Model and Problem Description

In this section, we mainly illustrate the system model, MAC protocol workflow and problem description.

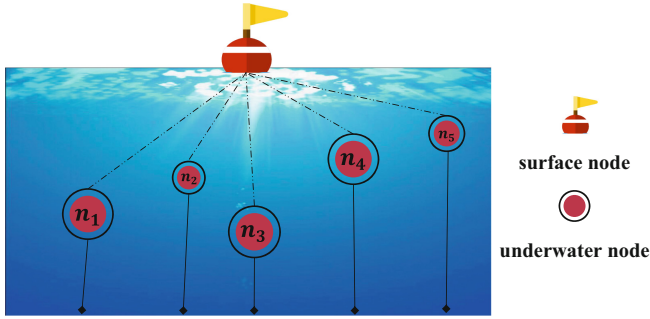


Fig. 1. The UASN model

2.1 Network Model

An UASN model with centralized topology is assumed, which includes a surface node SN and N underwater nodes $n = \{n_1, n_2, \dots, n_N\}$, as shown in Fig. 1. The SN can be charged by solar cells and mobile boats, so its energy is assumed to be unlimited. The underwater nodes are randomly deployed in a three-dimensional underwater area and anchored to the seabed using anchor chains. Each underwater node transmits data packets in a sequential cycle with time slots according to the time slot allocation table broadcasted by the SN. Normally, each underwater node is able to get in communication with the SN. Due to the current movement and ocean noise, the link quality between SN and underwater nodes may change dynamically.

According to the requirements of ocean applications, the packet size and packet generation time of various type of information data, such as numerical information and image information collected by underwater nodes are also

different. In the next time period, unknown changes may occur that cause different underwater nodes to change the type of information sensed and generate data packets of different sizes. In order for the SN to obtain the data packet situation in each underwater node’s queue for LSTM-DQN learning training, each underwater node needs to transmit a simple control packet to the SN before transmitting the data packet. The control packet contains only information about the data packet size $D_{i,k}^b$ and data packet generation time $D_{i,k}^t$ in the node queue. The notations related to the nodes involved in the network transmission process are described as in Table 1.

Table 1. Description of main notations

Parameter	Description	Parameter	Description
SN	Surface node	n_i	The i_{th} underwater node
$D_{i,k}$	The k_{th} data packet in the queue of n_i	$D_{i,k}^b$	The data packet size of $D_{i,k}$
$D_{i,k}^t$	The generation time of $D_{i,k}$	$D_{i,s}$	The number of time slots to n_i
D_i^K	The number of data packets in n_i	D_i^B	The sum of data packet sizes in n_i
D_i^V	The data packet generation rate of n_i	A_w^B	The total data packet size matrix
A_w^S	The number of allocated time slots matrix	A_w^K	The number of data packets matrix
F	The number of frames in window period	A_w^V	The data generation rate matrix

2.2 MAC Protocol Workflow

In the UASN, a TDMA-MAC protocol frame in which the number of time slots can be dynamically allocated is constructed. According to the usage of time slot allocation tables, we describe the work of the TDMA-MAC protocol of UASN by dividing it into three stages: A-network initial stage, B-time slot allocation stage, C-allocation table update stage.



Fig. 2. The number of time slots used by each underwater node

A-network initial stage: It is the stage when UASN is just finished deploying under the water and starts working. The SN will broadcast the initial time slot allocation table to each n_i . Each n_i will be assigned a fixed number of time slots in the initial stage’s time slot allocation table. Each n_i transmits the data packets from its own queue in its own time slot sequentially.

B-time slot allocation stage: After the UASN is stabilized in the initial stage, the SN continuously collect control data packets from each n_i within a certain number of window periods. The window period can be set for different frame sizes depending on the needs of the work. As in Fig. 2, the number of time slots used by each n_i will increase, decrease or remain unchanged.

C-allocation table update stage: Depending on the needs of different works, new changes may also occur in the data generated and uploaded at each n_i . The SN will start the B stage of generating the new time slot allocation table when the update threshold is reached by observing the control data packets collected during the window period.

After the network initial stage, both the SN and n_i will keep working between stages B and C until the end of the UASN work, as in Fig. 3. Finally, the dynamic adjustment of the number of time slots is implemented.

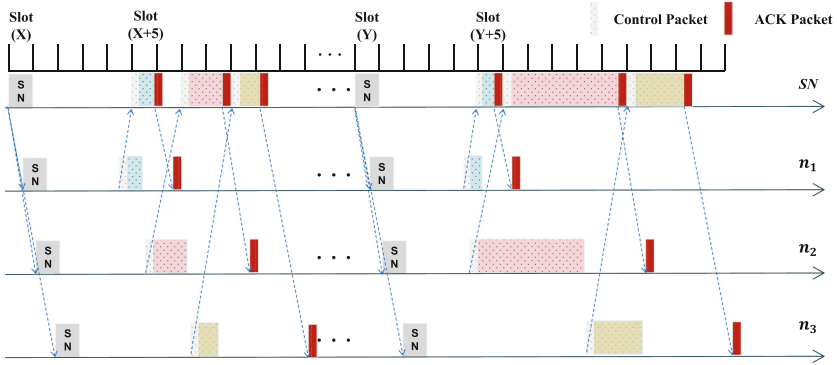


Fig. 3. The dynamic adjustment of the number of time slots

2.3 Problem Description

In the above TDMA-MAC protocol frame, the utilization efficiency of time slots of the UASN can be effectively improved by dynamically allocating the number of time slots for different underwater nodes. However, the time slot allocation strategy of TDMA-MAC protocol that cannot adapt to the packet generation rule (packet size and packet generation time) will lead to the following problems, which seriously reduce the network performance.

Case (1): An underwater node generates small data packets. The time used by the underwater node to transmit data packets is much less than the time allocated to it. This results in wastage and low utilization of time slots.

Case (2): An underwater node generates large data packets. The underwater node needs to split the data packet and wait for its own time slot time several times for transmission. Multiple transmissions generate multiple headers, resulting in extra transmission energy consumption and high end-to-end delay.

Case (3): A large number of information interactions between underwater nodes. The computational process of the protocol between neighboring nodes is complex, resulting in high interference and extra energy consumption.

Case (4): A large number of information interactions between underwater nodes and SN. The large underwater acoustic signal transmission delay and long node waiting time during the interaction increase the data packet end-to-end delay.

The design of the TDMA-MAC protocol in the UASN firstly needs to consider both problems in Case (1) and (2), so that nodes can fully utilize each time slot and improve throughput. It is also necessary to consider problems in Case (3) and (4), where control the extra energy consumption and reduce the end-to-end delay. In order to solve these problems, we design the time slot variable MAC protocol based on LSTM-DQN to flexibly allocate time slot table for each underwater node. The details of the strategy will be described in Sect. 3.

3 TSV-MAC Protocol

In our TSV-MAC protocol, the SN firstly stores the collected queue information of each n_i in the respective node information matrix after processing. Then, the SN predicts the generation and transmission of data by each n_i by using LSTM-DQN to learn the historical data in the n_i queue. Finally, the SN applies the prediction results to the update of the time slot allocation table.

3.1 Node Information Matrix

A node information matrix is designed to store the queue information of underwater nodes, which is used by the LSTM-DQN model to learn the historical data transmitted by underwater nodes. The node information matrices contain the number of allocated time slots matrix A_w^S , the number of data packets matrix A_w^K , the total data packet size matrix A_w^B , the data generation rate matrix A_w^V . These four information matrices clearly record the status of data packets in the queue from each underwater sensor node. Specifically, for the queue of each n_i , the SN needs to calculate the number of data packets D_i^K for A_w^K , the sum of data packet sizes D_i^B for A_w^B and the data packet generation rate D_i^V for A_w^V , respectively. Their mathematical expressions are as follows,

$$D_i^K = D_{i,k_M} - D_{i,k_0} + 1 \quad (1)$$

$$D_i^B = \sum_{m=0}^M D_{i,k_m}^b \quad (2)$$

$$D_i^V = \frac{D_i^B}{D_{i,k_M}^t - D_{i,k_0}^t} \quad (3)$$

In Eq. (1), D_{i,k_M} is the queue number of the last data packet, and D_{i,k_0} is the first data packet in the control data packet. In Eq. (2), D_{i,k_m}^b is the data packet size of the queue number through the record in control data packet. In Eq. (3), D_{i,k_M}^t is the generation time of the last data packet and D_{i,k_0}^t is the generation time of the first data packet in the control data packet.

The SN stores the information obtained by pre-processing in multiple information matrices. We set the size of each information matrix to $F * N$. F is the number of frames set in a window period. N is the number of underwater nodes in this UASN. Next, we illustrate the above four matrices with a window period of 10 frames, 5 underwater nodes, and a matrix size of $10 * 5$.

$$A_{18}^S = \begin{pmatrix} 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{pmatrix} \longrightarrow A_{18}^S = \begin{pmatrix} 1 & 6 & 3 & 2 & 2 \\ 1 & 6 & 3 & 2 & 2 \\ 1 & 6 & 3 & 2 & 2 \\ 1 & 6 & 3 & 2 & 2 \\ 1 & 6 & 3 & 2 & 2 \\ 1 & 6 & 3 & 2 & 2 \\ 1 & 6 & 3 & 2 & 2 \\ 1 & 6 & 3 & 2 & 2 \\ 1 & 6 & 3 & 2 & 2 \\ 1 & 6 & 3 & 2 & 2 \\ 1 & 6 & 3 & 2 & 2 \end{pmatrix}$$

Fig. 4. Examples of A_w^S

$$A_{18}^K = \begin{pmatrix} 2 & 18 & 6 & 3 & 4 \\ 2 & 15 & 7 & 3 & 4 \\ 1 & 10 & 5 & 3 & 3 \\ 3 & 6 & 5 & 2 & 2 \\ 2 & 8 & 5 & 4 & 4 \\ 2 & 22 & 5 & 2 & 3 \\ 1 & 18 & 8 & 3 & 2 \\ 2 & 13 & 5 & 2 & 1 \\ 2 & 19 & 6 & 1 & 2 \\ 1 & 10 & 6 & 2 & 2 \end{pmatrix} A_{18}^B = \begin{pmatrix} 32 & 265 & 87 & 43 & 59 \\ 33 & 221 & 103 & 41 & 57 \\ 18 & 167 & 74 & 40 & 43 \\ 45 & 83 & 73 & 29 & 32 \\ 34 & 115 & 69 & 58 & 58 \\ 31 & 328 & 75 & 30 & 43 \\ 17 & 267 & 118 & 45 & 28 \\ 32 & 193 & 76 & 31 & 16 \\ 31 & 285 & 89 & 17 & 29 \\ 16 & 147 & 84 & 32 & 30 \end{pmatrix} A_{18}^V = \begin{pmatrix} 2.1 & 9.4 & 6.9 & 2.1 & 2.1 \\ 2.2 & 9.7 & 7.3 & 2.7 & 2.7 \\ 1.0 & 9.0 & 4.3 & 2.7 & 1.9 \\ 3.2 & 5.9 & 5.2 & 2.0 & 2.0 \\ 1.4 & 8.2 & 4.9 & 1.4 & 4.1 \\ 1.9 & 23.4 & 5.3 & 0.9 & 1.8 \\ 1.1 & 8.5 & 8.4 & 3.2 & 1.8 \\ 2.3 & 7.6 & 3.4 & 1.9 & 1.2 \\ 2.0 & 19.4 & 6.3 & 1.2 & 2.0 \\ 1.0 & 3.0 & 6 & 2.3 & 2.1 \end{pmatrix}$$

Fig. 5. Examples of A_w^K , A_w^B , A_w^V

- (1) The matrix of the number of allocated time slots $A_w^S = (f, i)$, $f \in F$, $i \in N$, $(f, i) = D_{i,S}$, indicates at the $[(w-1) * 10 + f]$ frame, the number of time slots allocated for the n_i node according to the time slot allocation table. As in Fig. 4, in the initial stage, during the first window period, each n_i is allocated 2 time slots. After 16 window periods (160 frames), a new time slot allocation table is enabled. The new time slot allocation is recorded in the 18th window period. $(8, 3) = 3$ of A_{18}^S means that at frame 178, the n_3 node is allocated 3 time slots according to the time slot allocation table.
- (2) The number of data packets matrix $A_w^K = (f, i)$, $f \in F$, $i \in N$, $(f, i) = D_i^K$, indicates the number of data packets in the queue of n_i nodes at frame $[(w-1) * 10 + f]$. As in Fig. 5, $(8, 3) = 5$ of A_{18}^K indicates that there are 5 data packets in the queue of n_3 node at frame 178.
- (3) The total data packet size matrix $A_w^B = (f, i)$, $f \in F$, $i \in N$, $(f, i) = D_i^B$, indicates the total data packet size in the queue of n_i nodes at frame $[(w-1) * 10 + f]$.
- (4) The data generation rate matrix $A_w^V = (f, i)$, $f \in F$, $i \in N$, $(f, i) = D_i^V$, indicates the data packet generation rate in the queue of n_i nodes at frame $[(w-1) * 10 + f]$.

3.2 LSTM-DQN for TSV-MAC Protocol

In TSV-MAC protocol, we expect to predict the generation rule of data packets and allocate time slots through a deep reinforcement learning model, so as to reduce the information interaction between nodes and improve the utilization of time slots. In the process of time series prediction, the problem of long-term dependence is easy to appear, so that the dependent relevant information far

from the prediction point can not be used effectively. Here, we adopt LSTM to solve this problem, which allows the network to efficiently deal with the dependence of the current prediction on previous information and can avoid problems such as gradient disappearance of conventional Recurrent Neural Network. Meanwhile, we choose to use DQN algorithm in our reinforcement learning approach. Specifically, the definition of agent, action, state, and reward function in the LSTM-DQN algorithm is as follows:

- (1) Agent: The SN is used as agent because it can more conveniently charge and obtain time slot information than underwater nodes.
- (2) Action: The SN performs the action of increase or decrease the number of time slots by one for n_i in the time slot allocation table. If the number of time slots is 1 before decreasing, the action of no change is executed without decreasing. The action value a_{n_i} is *Add*, *Keep* or *Reduce*. These three values mean the number of time slots is increased by one, remained unchanged or decreased by one respectively.
- (3) State: After executing an action, we observe the situation in the node information matrix during the next window period. The node information matrix includes A_w^S , A_w^K , A_w^B and A_w^V .
- (4) Reward: Let the action be executed at the end of f -frame. We let T be a time slot unit time, $T'_i(j)$ be the usage time within the last time slot j allocated to node n_i after f -frame. The reward $r_i^1(j)$ and $r_i^2(j)$ are set by accumulating the utilization of node n_i in $T'_i(j)$ within a window period (let a window period be 10 frames), as follows:

$$R_{n_i} = \begin{cases} \sum_{j=1}^{10} r_i^1(j) & \text{if } a_{n_i} = \textit{Add} \\ \sum_{j=1}^{10} r_i^2(j) & \text{if } a_{n_i} = \textit{Keep} \text{ or } \textit{Reduce} \end{cases} \quad (4)$$

$$r_i^1(j) = \begin{cases} \frac{T'_i(j)}{T} & \text{if } T'_i(j) \neq 0 \\ -1 & \text{if } T'_i(j) = 0 \end{cases} \quad (5)$$

$$r_i^2(j) = \begin{cases} 1 - \frac{T'_i(j)}{T} & \text{if } T'_i(j) \neq T \\ -1 & \text{if } T'_i(j) = T \end{cases} \quad (6)$$

As in Fig. 6, the reward for an underwater node is described as an example. At the f -frame, the node is assigned 4 time slots. If the action to increase the number of time slots by 1 is executed at the end of the f -frame. We add up the time utilization of the node in the 5th time slot for the next 10 frames. If the node does not use the 5th time slot assigned to it, then the utilization is not 0, but -1 . This cumulative value is the reward. The higher this cumulative time slot utilization is, the higher the reward value will be.

On the contrary, reducing the number of time slots by 1 is executed. If the node fully uses the 3rd time slot allocated to it, then the nonavailability is not 0, but -1 . The higher this cumulative time slot nonavailability is, the higher the reward value will be. The same is true if the keep action is executed.

- (5) To reduce the energy consumption of the algorithm, we use a flexible training mechanism on SN. The reward difference U of 10 adjacent window periods is compared to determine whether the LSTM needs to be trained or not by falling below a threshold value.

3.3 Design of Time Slot Allocation Table

After the above calculation, the TSV-MAC protocol allocates a number of time slots to each n_i . We design a reasonable time slot allocation table for the protocol. In Fig. 7, the time slot allocation table, besides stating the number of time slots allocated to each n_i , also specifies the serial number of the time slot allocation table, the starting frame time for using this table and the total number of time slots allocated by this table.

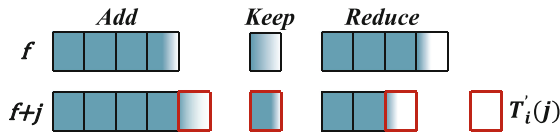


Fig. 6. The time used within the last time slot

Slot allocation table No.S:	Start Frame:	$D_{1,s}$: (number of time slots allocated to n_1)	$D_{2,s}$: (number of time slots allocated to n_2)	$D_{i,s}$: (number of time slots allocated to n_i)	$D_{N,s}$: (number of time slots allocated to n_N)	$\sum_{i=1}^N D_{i,s}$: (Total of time slots allocated by this table)
-----------------------------	--------------	--	--	-------	--	-------	--	--

Fig. 7. The time slot allocation table

Due to the complex underwater environment, the underwater nodes occasionally lose connection with the SN. SN will reduce the number of time slots used by this node. No matter how many time slots are reduced, the time slot allocation table will eventually reserve one time slot for it. When the n_i resumes communication with the SN, it transmits control data packets and data packets on its own time slot according to the latest time slot allocation table received. Then, the SN will add another number of time slots for its use.

4 Performance Evaluation

In this section, two typical MAC protocols TDMA and Slotted ALOHA are compared with our TSV-MAC protocol. The process of simulation experiments is implemented in python.

4.1 Simulation Scenario and Settings

In the UASN, underwater nodes are randomly deployed in underwater area. The network related parameters are shown in Table 2. Underwater nodes generate different numerical data packets, image data packets or audio data packets in each time slot in a Poisson distribution with parameter λ . We design two representative scenarios for simulation, which are described as follows:

- (1) High-load scenario: It is a scenario where the UASN is working at a busy time and continuously collecting data. The generation of all data packets obeys Poisson distribution, and the numerical data packets, image data packets and audio data packets correspond to the parameter $\lambda = 1, 1, 0.1$ respectively. In the high-load scenario, nodes sensing numerical information generate an average of 1 numerical data packet, sensing image information generate an average of 1 image data packet, and sensing audio information generate an average of 0.1 audio data packets per time slot.
- (2) Low-load scenario: It is a scenario where the UASN collects data at a low frequency when it is working idle. The generation of all data packets obeys Poisson distribution, and the numerical data packets, image data packets and audio data packets correspond to the parameter $\lambda = 1, 0.1, 0.05$ respectively.

In the TDMA and TSV-MAC protocols, if a data packet has been sent for more than 100 time slots but no ACK data packet is received, we consider it lost. In the Slotted ALOHA protocol, if there is a collision in data packet delivery, the data packet is randomly backed up by 1–10 time slots and resent.

Table 2. Simulation parameters

Parameter	Value	Parameter	Value
Deployment area size	1 km * 1 km * 1 km	N	3
Coordinates of SN	(500 m, 500 m, 0 m)	Numerical package size	2 kb
Time slot size	1 s	Image package size	50 kb
Transmit speed	100 kb/s	Audio package size	300 kb

4.2 Simulation Results

The simulation results are set as the average of 400 times. The performances of all schemes are evaluated by the following two metrics:

Throughput: The total amount of various data packets collected by the SN in a given period of time.

End-to-end delay: The queuing delay, which is the time data packets spend in the queue waiting to be sent out, is adopted to approximate the end-to-end delay of the network. Because the processing delay, sending delay and propagation delay can be regarded as the same between the same scenario and different MAC protocols and the main impact on end-to-end delay is queuing delay.

Figure 8 shows the learning process of the LSTM-DQN algorithm, the performance of the three protocols in terms of throughput and queuing delay under different load scenarios. From Fig. 8(a), we can see that the model was trained for a total of 500 episodes. We can see that the reward curves for the first 400 episodes fluctuate considerably, indicating that the algorithm is still trying to learn the optimal solution from the historical information. At this point, the LSTM-DQN algorithm has not yet converged. And after 400 episodes shows the process of convergence of the algorithm towards a stable reward value. There are some small fluctuations because the model is still trying to find the reward.

In Fig. 8(b), the performance of throughput is compared. In High-load, the TSV-MAC protocol allocates multiple consecutive time slots to nodes with large data packets, reducing the generation of data packet headers compared to TDMA. Slotted ALOHA experiences a large number of collisions, resulting in reduced throughput. In low-load, the TSV-MAC protocol allocates fewer time slots to each node. However, when occasionally a node has a large data packet, the TSV-MAC protocol adds a few time slots for it and then reduces them back later. Slotted ALOHA does not have as large a performance difference in throughput as at high load because there are no large numbers of collisions generated.

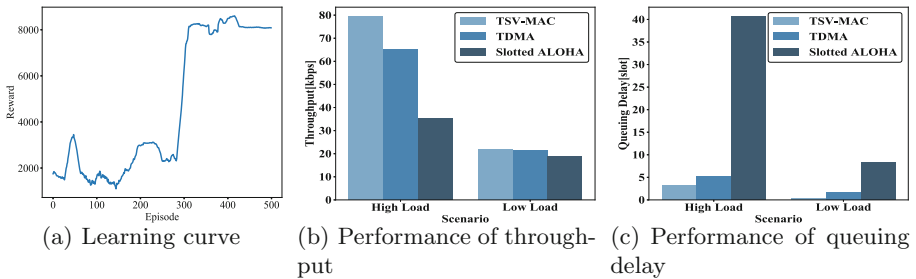


Fig. 8. Simulation results

In Fig. 8(c), the performance of queuing delay is compared. In High-load, the TSV-MAC protocol allows data packet-rich nodes to send in multiple consecutive time slots, reducing the queuing time of data packets in the queue of data packet-rich nodes compared to TDMA. In low-load, the TSV-MAC protocol adds time slots to the nodes with more occasional data packets, while other nodes may not have generated data packets yet. Therefore the impact on the queuing delay of other nodes is minimal. In the TDMA, nodes with a large number of data packets have to wait several times before they can finish sending them, thus the queuing delay is high. Slotted ALOHA also has the potential for collisions, and each collision adds a certain amount of queuing delay.

5 Conclusions

In UASNs of transmitting heterogeneous ocean data, we proposed TSV-MAC protocol based on LSTM-DQN to dynamically allocate time slots for underwater nodes. The TSV-MAC protocol fully considers the influence of the huge difference in data packet size generated by underwater nodes on time slot utilization, so that the SN can learn the historical data in the queue of each underwater node, and predict the data perception and transmission of each underwater node. The simulation results show that TSV-MAC can effectively improve the network throughput and reduce the queuing delay compared with the other two typical MAC protocols.

Acknowledgement. This research was supported in part by the National Natural Science Foundation of China-Guangdong Joint Fund under Grant No. U1701263, the National Natural Science foundation of China (NSFC) under grant No. 61871286, No. 61872266, No. 61672131, the Development Foundation of Tianjin Transportation Science and Technology under grant No. 2018-30, the Innovation Foundation of Tianjin University and Tianjin Key Laboratory of Advanced Networking (TANK).

References

1. Qiu, T., Zhao, Z., Zhang, T., Chen, C., Chen, C.L.P.: Underwater internet of things in smart ocean: system architecture and open issues. *IEEE Trans. Industr. Inform.* **16**(7), 4297–4307 (2020)
2. Zhao, Z., Liu, C.F., Qu, W.Y., Yu, T.: An energy efficiency multi-level transmission strategy based on underwater multimodal communication in UWSNs. In: *IEEE Conference on Computer Communications*, pp. 1579–1587 (2020)
3. Jiang, S.M.: State-of-the-art medium access control (MAC) protocols for underwater acoustic networks: a survey based on a MAC reference model. *IEEE Commun. Surv. Tutor.* **20**(1), 96–131 (2018)
4. Acar, G., Adams, A.E.: ACMENet: an underwater acoustic sensor network protocol for real-time environmental monitoring in coastal areas. *IEE Proc.-Radar Sonar Navig.* **153**(4), 365–380 (2006)
5. Nguyen, T.H., Shin, S.Y., Park, S.H.: Efficiency reservation mac protocol for underwater acoustic sensor networks. In: *Fourth International Conference on Networked Computing and Advanced Information Management*, pp. 365–370 (2008)
6. Lin, W., Li, D., Jian, C., Tao, S., Teng, W.: A wave-like amendment-based time-division medium access slot allocation mechanism for underwater acoustic sensor networks. In: *International Conference on Cyber-Enabled Distributed Computing & Knowledge Discovery*, pp. 369–374 (2009)
7. Cho, H.J., Namgung, J., Yun, N., Park, S.H., Kim, C.H., Ryuh, Y.S.: Contention free MAC protocol based on priority in underwater acoustic communication. In: *OCEANS 2011 IEEE Spain*, pp. 1–7 (2011)
8. Jia, M., Chen, S., Liu, Y., Yu, J., Xu, Y.: LT-MAC: a location-based TDMA mac protocol for small-scale underwater sensor networks. In: *IEEE International Conference on Cyber Technology in Automation*, pp. 1275–1280 (2015)
9. Lu, H., Feng, H., Guo, Z., Yang, X.: A TDMA-based MAC protocol in underwater sensor networks. In: *4th International Conference on Wireless Communications, Networking and Mobile Computing*, pp. 1–4 (2008)

10. Kredo, K., Djukic, P., Mohapatra, P.: STUMP: exploiting position diversity in the staggered TDMA underwater MAC protocol. In: IEEE Conference on Computer Communications, pp. 2961–2965 (2009)
11. Kredo, K., Mohapatra, P.: Scheduling granularity in underwater acoustic networks. In: Proceedings of 6th ACM International Workshop Underwater Networks, pp. 7:1–7:8 (2011)
12. Shahabudeen, S., Chitre, M., Motani, M.: MAC protocols that exploit propagation delay in underwater networks. In: OCEANS 2011 MTS, pp. 1–6 (2011)
13. Chen, Y.D., Lien, C.Y., Chuang, S.W., Shih, K.P.: DSSS: a TDMA-based mac protocol with dynamic slot scheduling strategy for underwater acoustic sensor networks. In: OCEANS 2011 IEEE Spain, pp. 1–6 (2011)
14. Ye, X.W., Yu, Y.D., Fu, L.Q.: Deep reinforcement learning based mac protocol for underwater acoustic networks. *IEEE Trans. Mob. Comput.* **21**(5), 1625–1638 (2022)
15. Park, S.H., Mitchell, P.D., Grace, D.: Reinforcement learning based mac protocol (UW-ALOHA-Q) for underwater acoustic sensor networks. *IEEE Access* **7**, 165531–165542 (2019)
16. Valerio, V.D., Petrioli, C., Pescosolido, L., Schaar, M.V.D.: A reinforcement learning-based data-link protocol for underwater acoustic communications. In: The 10th ACM International Conference on Underwater Networks & Systems (2015)
17. Sainath, T.N., Vinyals, O., Senior, A., Sak, H.: Convolutional, long short-term memory, fully connected deep neural networks. In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 4580–4584 (2015)



Localization for Underwater Sensor Networks Based on a Mobile Beacon

Ying Guo^(✉), Longsheng Niu, Rui Zhang, Hongtang Cao, and Jingxiang Xu

Qingdao University of Science and Technology, Qingdao 266100, Shandong, China
guoying@qust.edu.cn

Abstract. In Underwater Sensor Networks (UWSNs), the location information of sensor nodes is essential for making the measured data meaningful. However, UWSNs have a complex node deployment environment. Node mobility caused by ocean currents and other factors would lead to a bigger ranging error and make some nodes cannot receive enough data packets. In this paper, a Localization algorithm based on a Single Mobile Beacon (LSMB) is proposed. LSMB makes use of the attenuation law of signal strength and the geometric relationship between a sensor node and the path of the mobile beacon, reducing the impact of random error on distance measurement. On this basis, by analyzing the overall movement trends of sensor nodes, this paper analyzes and studies the counter-current movement and downstream movement of the mobile beacon respectively, so as to make LSMB suitable for dynamic marine environment. The simulation shows that the algorithm reduces the impact of node mobility on localization and has small average localization error.

Keywords: Underwater sensor networks · Node localization · Mobile beacon · Ocean current direction

1 Introduction

With the development of marine research, underwater wireless sensor networks (UWSNs) have attracted more and more attention [1]. The emergence of UWSNs provides a new platform for under communication to explore the underwater environment [2]. The location information of nodes is the basis for making the monitoring data meaningful [3], and it is also a prerequisite for network routing and coverage [4]. UWSNs contains beacon nodes and sensor nodes, and the beacon nodes can assisting sensor nodes in completing their localization [5,6]. Underwater sensor nodes use acoustic signals for communication [7], so the propagation delay is large and the bandwidth is limited, which brings a new challenge to the design of the localization method. Beacon nodes have a great impact on localization accuracy, but they are difficult to deploy accurately in the underwater environment. By observing the network that we deployed in Qingdao Bay,

we found that the data collection should be carried out through a ship, and the ship can get the accurate position through GPS or other techniques. Deploying a beacon node on the bottom of the ship not only reduces the deployment cost but also improves localization accuracy.

In Marine environment, the sensor nodes are randomly moved by the ocean currents, tides and other factors, which resulting a large ranging error [8,9]. This paper proposes a Localization algorithm based on a Single Mobile Beacon (LSMB). A beacon node fixed to a ship moves along a certain path and sends a series of beacon packages at a fixed interval. Each node passively receives signals and calculates its coordinates according to the geometric relations to the movement path of the beacon. And combined with the ocean current direction of the deployment region, different localization strategies are discussed.

The rest of this paper is organized as following. In Sect. 2, we describe the design of LSMB in both static and dynamic scenes. Following that, we present simulation results in Sect. 3. Finally, we conclude the paper in Sect. 4.

2 Localization Algorithm Design

2.1 LSMB in Static Environment

The ship moves along a straight line, and the beacon node fixed to it sends location package to underwater nodes at regular intervals, in which includes the real-time position of the beacon node and the initial energy and frequency of the signal. In wireless communication, the bigger the distance, the greater the signal propagation loss. An underwater sensor node could use the received signal strength to calculate the distance to these beacon points.

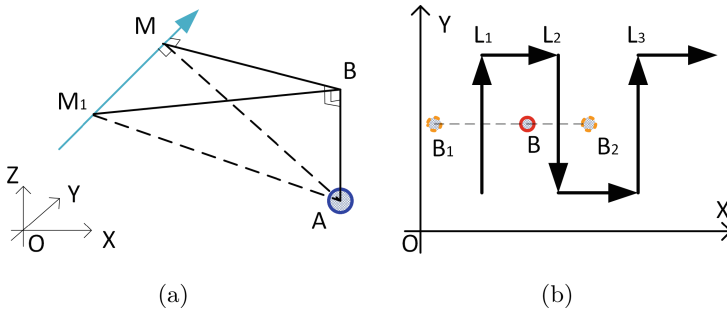


Fig. 1. (a) The distance calculation between a sensor node and the trajectory of the mobile beacon, (b) the SCAN trajectory

As shown in Fig. 1(a), there is an underwater sensor node A and point B is its projection position at the sea level. M_1 is a beacon point on the trajectory of the beacon node, which could communicate with node A . Make a perpendicular

from node A to the trajectory of the beacon node, and the foot point is M , and connect M_1A , MA , M_1B , and MB . The depth of the node is obtained from a pressure sensor, so the length of AB is known. According to the geometric relations, the length of MB can be calculated.

$$MB = \sqrt{M_1A^2 - AB^2 - M_1M^2} \quad (1)$$

In Eq. (1), the length of M_1M is unknown. Assume M' is the closest beacon point on the line to node A , so M' is regarded as the foot point M . Then the value of M_1M is an integral multiple of the beacon sending interval. According to the number of beacon points between M_1 and M' , the value of M_1M is known. Thus, the value of MB can be obtained.

Similarly, the length of MB can also be calculated according to other beacon points on the line. The average of these calculations is taken as the final result of MB . Assume that the beacon moves along the direction perpendicular to the X-axis, then the coordinate of node A can be calculated by Eq. (2).

$$\begin{cases} x_A = x_M \pm MB \\ y_A = y_M \\ z_A = z_M - h \end{cases} \quad (2)$$

It can be observed that there are two possible solutions. For eliminating the wrong coordinates among them, the beacon node moves along the SCAN path [10], as shown in Fig. 1(b), which includes main trajectory and auxiliary trajectory. Main trajectory is perpendicular to X-axis, and is used for distance measurement. Auxiliary trajectory is perpendicular to the Y-axis, and is used for changing direction. As shown in Fig. 1(b), through L1, a node could know that it is on the left or right side of L1, and through L2, it could know that it is on the left or right side of L2. Thus, through two adjacent main trajectories, the node can localize itself eventually.

The length of the auxiliary trajectory is related to the number of packets that the underwater sensor node needs to receive on the main trajectory. If each node needs to receive at least m consecutive packets on every main trajectory, the length of the auxiliary trajectory should meet.

$$H_1 \leq \sqrt{R^2 - \left(\frac{m}{2} \times VT\right)^2 - h^2} \quad (3)$$

where R is the communication range of the beacon node. And in the design of LSMB, we set the value of m is 5.

2.2 LSMB in Dynamic Environment

From [11, 12], underwater nodes affected by ocean currents and tides have obvious movement trend. Here we define the moving direction of the beacon node on the auxiliary trajectory as the scanning direction. By the example of the trajectories

in Fig. 1(b), the scanning direction is from left to right. When an underwater node moves along with the ocean current, it receives packets sent by the beacon. In addition to the beacon position and trajectory information, each packet also contains the movement trend. The trend is a boolean variable, 0 and 1 represents left or right respectively. If the scanning direction of the beacon node is opposite to the movement trend of underwater nodes, it is called counter-current scanning, otherwise it is called downstream scanning.

In a dynamic environment, there may be a mirror image error. For solving this problem, we need to modify the length of the auxiliary trajectory. Obviously, the length of the auxiliary trajectory is related to whether the beacon node scans in the counter-current direction or downstream direction. The localization strategies under these two scanning modes are discussed below.

Counter-Current Scanning. Assume the beacon node is from left to right and all unlocated nodes are moved from right to left. As shown in Fig. 2(a), the sub-region between L_2 and L_3 could be divided into left and right parts by line L_0 . The nodes in the left part may have a mirror image error, but the breadth of the left part is small, so that the absolute value of error is small. Whereas the nodes in the right part cannot receive signals from L_1 , and because they move from right to left, they must be on the left side of L_3 . So there is no mirror image error at this time. For other sub-region in the whole deployment region, it has the same result. The left area is much smaller than the right area, so the mirror error of the left part is ignored.

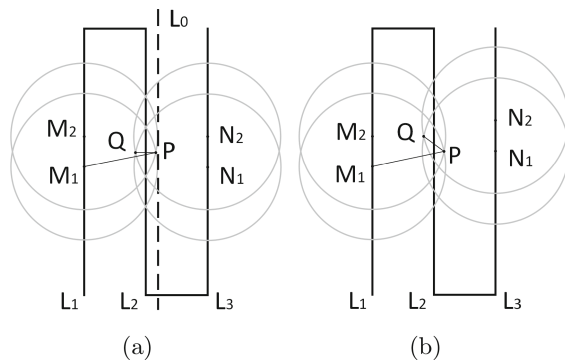


Fig. 2. (a) Node crossing, (b) node crossing on a slant

Because nodes are moving, a small number of nodes may move from one side of a main trajectory to the other, and the movement of the beacon just makes these nodes cannot be located. This problem is called node crossing in this paper. Taking Fig. 2(a) as an example, M_1 and M_2 are two adjacent beacon points on L_1 , N_1 is a beacon point on L_3 , and each of the three points correspond to a communication range (communication circle). P , Q are two intersections of two

communication circles. Suppose there is a node on the right side of point P , which is outside the coverage area of all beacon points on L_1 . After a period of time, the beacon node travels to N_1 , and the node has moved to the left of point Q . At this time, the node is outside the coverage area of all points on L_3 . Therefore, the node can only receive m packets from L_2 , and the localization fails. This problem limits the value of the auxiliary trajectory length.

In extreme cases, the moving direction of the underwater node is completely opposite to the scanning direction of the beacon node. At this time, the amount of node movement causing this problem is the smallest. The x-coordinate of point P is the smallest x-coordinate of the uncovered part of trajectory L_1 , and correspondingly, the x-coordinate of point Q is the biggest x-coordinate of the uncovered part of trajectory L_3 . Therefore, the line segment PQ corresponds to the minimum movement distance for the problem to occur.

Suppose that the line equation of L_1 is $x = i$, the x-coordinate of point P is p . Then the distance between point P and L_1 is

$$p - i = \sqrt{R^2 - \left(\frac{VT}{2}\right)^2} - h^2 \tag{4}$$

Obviously, point Q is the mirror position of point P with respect to line L_1 , and the length of PQ is

$$PQ = 2(p - i - H_2) \tag{5}$$

where H_2 is the length of the auxiliary trajectory. From point M_1 to point N_1 , the beacon travels about $2(L + H_2)$, where L is the length of the main trajectory. The time consumption of underwater nodes in PQ shall be greater than the time consumption of the beacon node from M_1 to N_1 .

$$\frac{PQ}{v} \geq \frac{2(L + H_2)}{V} \tag{6}$$

where v is the moving speed of the underwater node, from Eq. (4), Eq. (5), and Eq. (6), for avoiding the node crossing, H_2 should meet.

$$H_2 \leq \frac{V\sqrt{R^2 - \left(\frac{VT}{2}\right)^2} - h^2 - vL}{V + v} \tag{7}$$

When the moving direction of a sensor node is not completely opposite to the scanning direction, the nearest beacon points M_1 and N_1 are at different y-coordinate. As shown in Fig. 2 (b), it is easy to know that the lateral distance between the point P and the point Q is equal to the length of PQ in Fig. 2(a). Through the Pythagorean theorem, compared with Fig. 2(a), the length of PQ in Fig. 3(b) is more larger. Thus, it has a fewer restriction on the auxiliary trajectory length, so we would not discuss it in this paper.

If a node can receive 5 data packets from two main trajectories respectively, it is more appropriate to use the beacon points on the right trajectory for localization calculation, because the time interval from beacon sending time to the

calculation time is smaller and the impact caused by node movement is small. If the number of data packets from the right trajectory is less than 5, the left trajectory is used for localization calculation, whereas the right trajectory is for eliminating the wrong position.

Downstream Scanning. When the beacon is from left to right and sensor nodes are also from left to right, assume that the beacon could start scanning from the outside of the deployment region. It is well known that if a node moves from left to right, then the first trajectory it heard must be on the left side of the node (If the node moves from right to left, this mode determines the beacon also moving from right to left, the result is the opposite). And the first heard trajectory could be regarded as the left main trajectory for localization calculation.

The above processes completely eliminate the mirror image error. However, the node crossing also exists in this mode. It occurs when the lateral movement of underwater nodes is larger than the beacon. So the length of the auxiliary trajectory should meet.

$$\frac{H_3}{v} \geq \frac{2(L + H_3)}{V} \quad (8)$$

$$H_3 \geq \frac{2vL}{V - 2v} \quad (9)$$

Equation (9) limits the lower bound of the auxiliary trajectory. As for its upper bound, the length should ensure that a node can receive at least m packets from the remote main trajectory, that is, it should be less than H_1 . If the speed difference between the beacon node and the underwater sensor node is too small, the lower and upper bound of the auxiliary trajectory length cannot meet at the same time. At this time, we can reduce m or V . From the perspective of localization accuracy, the latter is chosen here, and the upper bound is H_1 .

3 Simulations

3.1 Settings

We used Matlab R2020a to evaluate the performance of LSMB. And in the future, we would gradually carry out ocean experiments. 300 sensor nodes are deployed in the range of 1000 m * 1000 m * 100 m. The signal frequency is set to 40 kHz, and the absorption loss coefficient n is 1.5. Assume the error of the sonar device obeys a normal distribution with zero as its mean value and 0.5 as its standard deviation. The speed of the beacon node is 5 m/s. Average localization error is used to judge the stability of algorithm. The localization error of a sensor node is defined as the Euclidean distance between the estimated position and the real position. The algorithm proposed in this paper is compared with the algorithm (MANARL) proposed in [13] and a least squares localization method based on SCAN path (SCAN-LS) [10]. SCAN-LS applies the least squares method to the SCAN path and it does not utilize the movement trajectory information of the mobile beacon.

3.2 Simulations for LSMB

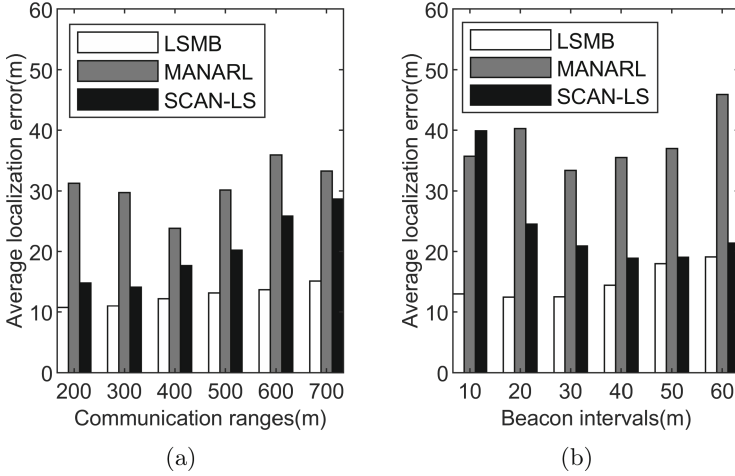


Fig. 3. Average localization errors under: (a) different communication ranges, (b) different beacon sending intervals

Figure 3 indicates the influence of communication range and beacon sending interval on average localization error in the static environment. MANARL does not completely eliminate the mirror error, so its localization error is hard to reduce through changing communication range and beacon sending interval. The localization error of SCAN-LS is obviously improved with the increase of communication range. When the beacon sending interval is small, the average localization error of SCAN-LS is larger because of the collinearity of beacon points in the multilateration method.

Figure 4 compares the localization performance of the two scanning modes in different communication ranges and different movement speeds. In each second, underwater nodes drifting in a random direction in the range of $[\pi/2, \pi]$ (i.e., the counter-current scanning of the beacon) and $[0, \pi/2]$ (i.e., the downstream scanning of the beacon) are simulated respectively. The beacon sending interval is set to 30 m. It shows that there is little difference in the two scanning modes on the average localization error.

In general, with the increase of movement speed of underwater nodes, the localization error of LSMB is improved. One exception is when the communication range is 200 m, and the reason is that the counter-current localization does not completely solve the problem of mirror error. When the communication range is small, the trajectory density of SCAN path is large, and more nodes across the line, so there is a greater possibility of producing mirror error during the localization process, resulting in the instability of the localization results.

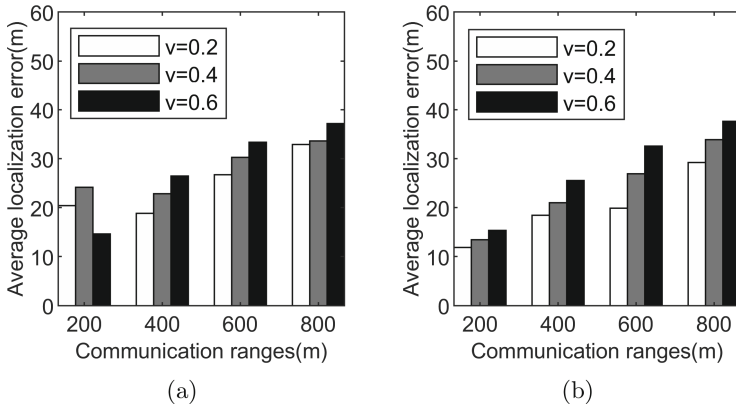


Fig. 4. (a) Localization errors while the beacon moving counter-current; (b) localization errors while the beacon moving downstream

4 Conclusion

This paper presents a novel node localization method based on a single mobile beacon for UWSNs. A beacon node moves along a SCAN trajectory and sends signals periodically. Sensor nodes use the change regularities of propagation loss and the geometric relations between the node and movement trajectories of the beacon to estimate their own positions. For dynamic ocean environment, according to a common tidal mobility model, the movement trend of underwater nodes is analyzed, then the beacon's counter-current scanning and downstream scanning are discussed respectively. Simulation results demonstrate that the proposed method has a relatively high localization accuracy and has fewer extreme localization errors, and that can suitable for both static and dynamic ocean environments.

Acknowledgement. This work was supported by Natural Science Foundation of Shandong Province (No. ZR2020MF061).

References

1. Su, Y., Guo, L., Jin, Z., Fu, X.: A mobile-beacon-based iterative localization mechanism in large-scale underwater acoustic sensor networks. *IEEE Internet Things J.* **8**(5), 3653–3664 (2021). <https://doi.org/10.1109/JIOT.2020.3023556>
2. Ullah, I.: A review of underwater localization techniques, algorithms, and challenges. *J. Sens.* **2020** (2020). <https://doi.org/10.1155/2020/6403161>
3. Yan, J., Meng, Y., Yang, X., Luo, X., Guan, X.: Privacy-preserving localization for underwater sensor networks via deep reinforcement learning. *IEEE Trans. Inf. Forensics Secur.* **16**, 1880–1895 (2021). <https://doi.org/10.1109/TIFS.2020.3045320>

4. Mridula, K., Ameer, P.: Localization under anchor node uncertainty for underwater acoustic sensor networks. *Int. J. Commun. Syst.* **31**, e3445 (2017). <https://doi.org/10.1002/dac.3445>
5. Karagol, S., Yildiz, D.: A path planning model based on nested regular hexagons using weighted centroid localization. *Int. J. Commun. Syst.* **35**(1), e5015 (2022). <https://doi.org/10.1002/dac.5015>
6. Huang, H., Zheng, Y.R.: Node localization with AOA assistance in multi-hop underwater sensor networks. *Ad Hoc Netw.* **78**, 32–41 (2018). <https://doi.org/10.1016/j.adhoc.2018.05.005>
7. Luo, J., Yang, Y., Wang, Z., Chen, Y.: Localization algorithm for underwater sensor network: a review. *IEEE Internet Things J.* **8**(17), 13126–13144 (2021). <https://doi.org/10.1109/JIOT.2021.3081918>
8. Osborn, J., Qualls, S., Canning, J., Anderson, M., Edwards, D., Wolbrecht, E.: AUV state estimation and navigation to compensate for ocean currents. In: *OCEANS 2015 - MTS/IEEE Washington*, pp. 1–5 (2015). <https://doi.org/10.23919/OCEANS.2015.7401906>
9. Bhairavi, R., Sudha, G.F.: Modified dive and rise technique incorporating enhanced weighted centroid localization algorithm with ocean current mobility model in underwater acoustic sensor networks. In: Ranganathan, G., Chen, J., Rocha, Á. (eds.) *Inventive Communication and Computational Technologies*. LNNS, vol. 89, pp. 569–582. Springer, Singapore (2020). https://doi.org/10.1007/978-981-15-0146-3_54
10. Koutsonikolas, D., Das, S.M., Hu, Y.C.: Path planning of mobile landmarks for localization in wireless sensor networks. *Comput. Commun.* **30**(13), 2577–2592 (2007). <https://doi.org/10.1016/j.comcom.2007.05.048>
11. Beerens, S., Ridderinkhof, H., Zimmerman, J.: An analytical study of chaotic stirring in tidal areas. *Chaos Solitons Fractals* **4**(6), 1011–1029 (1994). [https://doi.org/10.1016/0960-0779\(94\)90136-8](https://doi.org/10.1016/0960-0779(94)90136-8)
12. Caruso, A., Paparella, F., Vieira, L.F.M., Erol, M., Gerla, M.: The meandering current mobility model and its impact on underwater mobile sensor networks. In: *IEEE INFOCOM 2008 - The 27th Conference on Computer Communications*, pp. 221–225 (2008). <https://doi.org/10.1109/INFOCOM.2008.53>
13. Sun, Y., Yuan, Y., Xu, Q., Hua, C.: A mobile anchor node assisted RSSI localization scheme in underwater wireless sensor networks. *Sensors* **19**, 4369 (2019). <https://doi.org/10.3390/s19204369>

Vehicular Networks



Dataset for Evaluation of DDoS Attacks Detection in Vehicular Ad-Hoc Networks

Hong Zhong¹, Fan Yang¹, Lu Wei¹, Jing Zhang¹, Chengjie Gu²,
and Jie Cui¹(✉)

¹ Anhui University, Hefei, China
cuijie@mail.ustc.edu.cn

² New H3C Group, Hefei, China

Abstract. Vehicular ad-hoc networks (VANETs) are core components of the cooperative intelligent transportation system (C-ITS). Vehicles communicate with each other to obtain traffic conditions on the current road segment by broadcasting authenticated safety messages using their digital certificates. Although this method protects the system against external threats, it is ineffective when faced with internal adversaries who possess legal certificates. Consequently, an increasing number of researchers have focused on intrusion detection (misbehavior detection) technology. VeReMi and its extension version are the only public misbehavior datasets of VANETs in its field, allowing researchers to compare their studies with those of others. We note that denial of service (DoS) attacks in these datasets are insufficiently comprehensive. As a result, we designed a more complete dataset than existing datasets by implementing multiple attacks, including different types of distributed denial of service (DDoS) attacks. We present the detection results of some machine learning algorithms on our proposed dataset. These results indicate that our dataset can be utilized as a reference for future studies to evaluate different detection methods.

Keywords: Vehicular networks · DDoS · Dataset · Misbehavior detection

1 Introduction

VANETs form the core technology of cooperative intelligent transportation system (C-ITS) in future traffic management. They mainly consist of devices such as connected vehicles and roadside units (RSU). VANETs communicate road conditions to avoid collisions and reduce traffic congestion by exchanging safety messages. The Institute of Electrical and Electronics Engineers (IEEE) has developed a series of standards (IEEE 1609) for vehicles communication, of which

The work was supported by the National Natural Science Foundation of China (61872001), the Excellent Youth Foundation of Anhui Scientific Committee (2108085J31), and the Special Fund for Key Program of Science and Technology of Anhui Province, China (202003A05020043).

the IEEE 1609.2 specifies the cryptography method, public key infrastructure (PKI), for ensuring the security of vehicles messages. PKI distributes digital certificates to vehicles connected to the network. The vehicle that owns a certificate is regarded as a legitimate user, and it can encrypt and sign the message to be sent with a certificate to ensure the security and availability of the message. This method can effectively defend against attacks from outside the system. However, legitimate vehicles that have obtained certificates can still launch internal attacks. For example, they can send tampered messages or launch denial of service (DoS) attacks. Intrusion detection (misbehavior detection) is considered to be an effective method for defending against insider attacks. As a result, intrusion detection technology is becoming increasingly important to academics.

A dataset is crucial when deploying and comparing intrusion detection systems, particularly detection techniques based on machine learning methods. However, currently, there is a scarcity of datasets dedicated to vehicular networks. VeReMi [7] was the first public dataset for vehicle misbehavior detection. The dataset and its extended datasets, VeReMi Extension [11] and DARE [6], have made significant contributions to subsequent research to train their detection models and test performance. Although the aforementioned datasets cover a variety of attack models, there are no datasets for distributed denial of service (DDoS) attacks. Many researchers have not compared their studies [9, 12, 17] with others because of the lack of a reference DDoS attack dataset. This study fills the aforementioned gap.

In this study, we designed and published a DDoS attack dataset dedicated to vehicular networks. This dataset contains different types of DDoS attacks, including different attack frequencies (e.g., increasing rate and pulse attack) and different message contents. Compared to simple massive packet flooding attacks, these attacks are difficult to detect in time. The generation of this novel dataset is the main contribution of this study. Our second contribution is that we tested several machine learning algorithms on our dataset and compared their performance in detecting DDoS attacks, presenting the results in this paper for reference to other researchers.

The remainder of this paper is organized as follows: Sect. 2 discusses related works. Section 3 describes the communication and attacking models. In Sect. 4, we discuss the details of the simulations used to build the dataset. Section 5 presents the experimental results. Finally, we conclude the study in Sect. 6.

2 Related Works

Since the connected vehicles have not been deployed on a large scale, it is difficult to collect data of a large scale from the real scenes. The current international mainstream idea for creating datasets is to run simulation experiments utilizing a network simulation platform combined with a traffic flow simulation platform, then gather the data collected during the experiment to generate the dataset.

van der Heijden et al. [7] published a dataset called VeReMi as the first public dataset dedicated to misbehavior detection in VANETs using Veins framework combined with OMNeT++ and SUMO. They considered five types of attackers (tampering with position information in message), three different attacker probabilities, three different traffic densities, and five different random seeds. VeReMi dataset consists of 225 individual simulations, providing other researchers with an evaluation baseline.

Kamel et al. [11] upgraded VeReMi dataset by devising and implementing a realistic sensor error, which was added to the four main fields, position, speed, acceleration, and heading, in BSM. They also implemented a large more complex set of attacks, including data replay, DoS, and Sybil attacks. VeReMi Extension is more complete compared to its original version, but it still has a limitation in that it only supports local detection.

In order to solve this problem, Haidar et al. [6] published a dataset that supports global misbehavior detection. In their work, an individual vehicle performs simple plausibility checks and reports its result to the up architecture such as Misbehavior Authority [10] or other cloud components. The dataset consists of misbehavior report logs and original messages collected from vehicles, enabling other researchers to evaluate their cloud architecture or compare it to others.

There are also studies [1, 2, 5] that provided methods for generating a dataset in VANETs containing DoS or DDoS attacks without the use of Veins. See Table 1 for more information on above studies.

Table 1. Related works

	Tools	Standard	Attack frequency	Attackers collaboration	Multiple attacker density	Public
[7]	Veins	IEEE WAVE	Fixed	×	✓	✓
[11]	Veins	IEEE WAVE	Fixed	×	×	✓
[6]	Veins	IEEE WAVE	Fixed	×	✓	✓
[2]	ns-3	TCP/IP	Fixed	×	×	×
[1]	inet	UDP	Fixed	×	×	×
[9]	VimRTI	ETSI ITS	Fluctuation	×	×	×
Our dataset	Veins	IEEE WAVE	Fluctuation	✓	✓	✓

3 System Models

3.1 Communication Model

C-ITS is an upcoming technology for managing traffic and improving road safety in the future. C-ITS consists of different Intelligent Transportation System Stations (ITS-Ss), which can be On-Board Units (OBUs) embedded on vehicles or Road Side Units (RSUs). There are two communication models. According to IEEE 1609 standard, also known as IEEE Wireless Access in Vehicular Environments (WAVE) Architecture, the safety application mounted on the ITS-S can

send Basic Safety Messages (BSMs) and other safety messages to the surrounding ITS-Ss in the wireless environment through the WAVE protocol in the form of one-hop broadcast to exchange road condition information. BSM consists of information such as the real-time position and speed of the vehicle, which can assist autonomous driving, reduce congestion and avoid collisions. ITS-Ss can access the Internet through IPv6 and TCP/UDP protocols to obtain other services, such as navigation services and various entertainment services. This paper focuses on the first communication model, the communication between ITS-Ss. We only regard ITS-Ss as vehicles.

3.2 Attack Models

DoS attacks are designed to deplete the resources of victims. So that they cannot process legitimate service requests. One of the most common forms of attack is sending massive packets to the victim. DDoS attacks achieve this goal by deploying multiple attacking entities. Compared with the attacks of a single attacking entity, DDoS attacks are more threatening and the forms of attacks are more variable, making it easy to evade detection. A DDoS attack consists of several phases. The attacker first recruits multiple agent machines, which are controlled by the attacker, through scanning of remote machines or any other measures, looking for security holes that will enable subversion. Then, the attacker exploits the discovered vulnerability to break into recruited machines and infect them with attack code. Subverted agent machines will cooperate to launch an attack on the target in the final phase. In the vehicular environment, this paper implements a DDoS attack by flooding BSMs, the most frequent messages sent between vehicles. Due to the limitation of the simulation platform, we only simulate the final launch phase of a DDoS attack, and the first three phases are beyond the scope of this paper. We assume that the attacker somehow compromised some vehicles in the simulation area and ordered a DDoS attack at some point. According to the taxonomy of DDoS attack described in [15], combined with the characteristics of the vehicular networks, we selected four different attack forms for simulation:

- **Constant rate:** Controlled vehicles simultaneously send a large number of BSM packets to the victim at a constant rate, depleting a lot of resources and making it unable to process the BSMs sent by normal vehicles. There is a high possibility of causing a traffic accident due to the lack of BSMs received from normal vehicles.
- **Increasing rate:** The attacker starts sending BSMs to the victim at a lower frequency, then gradually reduces the sending interval, and finally reaches a higher sending frequency. This increasing rate attack is designed to evade detection or delay detected time. We considered two functions that shorten the BSM interval:

I_c	\triangleq	current beaconInterval
I_p	\triangleq	previous beaconInterval
r	\triangleq	beaconInterval increasing rate
$U(x, y)$	\triangleq	uniform distribution
dv_{min}	\triangleq	min drop value of interval
dv_{max}	\triangleq	max drop value of interval

1) Exponential function:

$$I_c = I_p \times r, r \in (0, 1) \tag{1}$$

2) Linear function

$$I_c = I_p - U(dv_{min}, dv_{max}) \tag{2}$$

- **Pulse attack/On-Off attack:** Controlled vehicles are divided into multiple groups. During a preset time period, the controlled vehicles in the same group simultaneously flood the victim with BSMs at a constant rate. After a period of time, the frequency of sending data packets from this group of vehicles returns to normal. The next group of attackers starts attacking. This alternate attack method brings great difficulty to detection. Victims can detect attacks, but it is difficult to detect all attackers at the first time. The current packet sending interval of the first group of attackers is calculated as follow:

I	\triangleq	normal beaconInterval
I_a	\triangleq	attacking beaconInterval
n	\triangleq	total number of groups
d	\triangleq	attacking duration per round

$$I_c = \begin{cases} I_a, & \cos(\frac{2\pi}{n \times d}x - \frac{\pi}{n}) - \cos(\frac{\pi}{n}) \geq 0 \\ I, & \cos(\frac{2\pi}{n \times d}x - \frac{\pi}{n}) - \cos(\frac{\pi}{n}) < 0 \end{cases} \tag{3}$$

n is the total number of groups and d is the duration of a single attack

- **Increasing-pulse:** This is a combination of the above two forms of attack. The controlled vehicles are also divided into multiple groups, and vehicles in the same group launch attack in the form of increasing rate. Different from pulse attack, this attack method does not start the next group after the previous group of attacks ends, but the next group starts to send BSMs to the victim at a low frequency after the attack has been carried out for a period of time. The traffic received by the normal vehicles have always remained at a high level. Such attack is more difficult to be detected because each group of attackers quickly returns to normal when the attack frequency peaks.

$$I_c = \begin{cases} \frac{I \times r^{x \bmod (n \times d)}}{r}, & \cos(\frac{2\pi}{n \times d}x - \frac{\pi}{n}) - \cos(\frac{\pi}{n}) \geq 0 \\ I, & \cos(\frac{2\pi}{n \times d}x - \frac{\pi}{n}) - \cos(\frac{\pi}{n}) < 0 \end{cases} \tag{4}$$

Figure 1 visually shows the regularity of the transmission frequency of BSMs sent by the attacker over time.

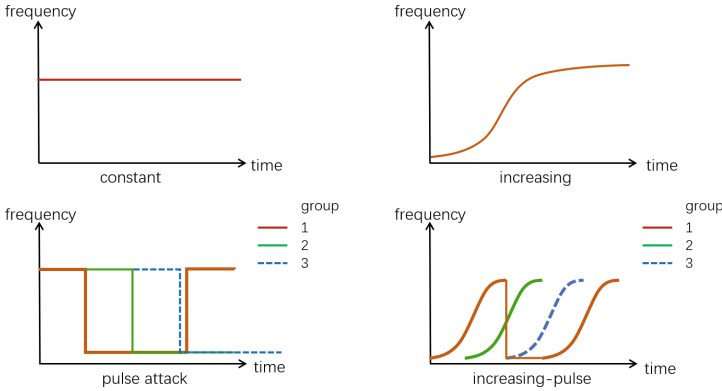


Fig. 1. Frequency of different DDoS attacks.

In addition to pure DDoS attacks, we also consider the case of attackers tampering with BSMs, combining it with DDoS attacks. Although large-traffic DDoS attacks can be quickly detected, BSMs sent in low frequency are treated as normal messages, which increases the possibility of causing traffic accidents. We followed [1] and [2] to implement attacks on tampering of the position, speed, and heading fields in BSM. The attacker adds a random offset to its true value in one of the three fields, which will cause the surrounding vehicles to misjudge the road conditions and cause collisions or other accidents.

– **Fake position:**

$$\begin{aligned}
 pos_x^f &= pos_x^r + U(R_{min}^{pos,x}, R_{max}^{pos,x}) \\
 pos_y^f &= pos_y^r + U(R_{min}^{pos,y}, R_{max}^{pos,y})
 \end{aligned}
 \tag{5}$$

– **Fake speed:**

$$spd^f = spd^r + U(R_{min}^{spd}, R_{max}^{spd})
 \tag{6}$$

– **Fake heading:**

$$\begin{aligned}
 h_x^r &= \cos(h^r) \\
 h_y^r &= -\sin(h^r) \\
 h_x^f &= h_x^r + U(R_{min}^{h,x}, R_{max}^{h,x}) \\
 h_y^f &= h_y^r + U(R_{min}^{h,y}, R_{max}^{h,y}) \\
 h^f &= atan2(h_y^f, -h_x^f)
 \end{aligned}
 \tag{7}$$

pos , spd , h are position, speed and heading respectively. r is real information in BSM and f is fake information. R represents the offset range. $U(x, y)$ is a uniform distribution. cos and sin are trigonometric functions. $atan2(y, x)$ is a function in C++ that return azimuth.

4 Simulation

4.1 Parameter Settings

We use the same tools as it is used to generate VeReMi, Veins framework [21] combining OMNeT++ [22] and SUMO [13], for code writing and simulation. Veins is an open-source vehicular network simulation framework written in accordance with the IEEE WAVE architecture, which is why we chose this framework for our experiments. OMNeT++ is an extensible, modular, component-based C++ simulation library and framework, primarily for building network simulators. SUMO is a mainstream traffic flow simulation platform that provides Veins with traffic data.



Fig. 2. Simulation area 1.

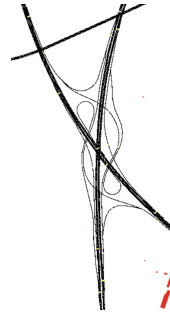


Fig. 3. Simulation area 2.

Like VeReMi and VeReMi Extension, we choose the Luxembourg SUMO Traffic (LusT) [4] scenario as the experimental scenario. The difference is that we did not select a large area for simulation. In this work, we focus on the implementation of DDoS attacks. Limited by the transmission power of the vehicle, the vehicle cannot receive messages from vehicles at a greater distance. In this case, attackers scattered in various places cannot launch coordinated attacks on the target vehicle. Therefore, we choose a smaller area for simulation. We select two areas as shown in Fig. 2 and Fig. 3: one is the urban traffic intersection, 5200, 6200–5900, 6800, with low vehicle speed; the other simulation area is a highway, 3600, 3500–4400, 5000, with high speed and high vehicles density.

Since launching a DDoS attack requires controlling multiple vehicles, it is difficult to launch a large-scale attack during a period of low traffic flow, so we choose the peak period (7 h) for simulation. We assume that at the beginning of the simulation, the attacker has taken control of some surrounding vehicles and is ready to attack. The simulation time is set to 75 s. In the first 15 s, all vehicles send normal messages. At the 15th second, some controlled vehicles launch a DDoS attack, and the attack time lasts for one minute. There are two reasons why the simulation time in this work is shorter than it in previous researchers' work: 1) A DDoS attack requires a coordinated attack by multiple controlled

vehicles. After driving for one minute, some attacking vehicles are likely to be far away from the target vehicle, and the attack effect will be seriously affected. It becomes a DoS attack launched by a scattered single vehicle. Researchers have done it. Similar scenarios can be found in the datasets of previous researchers. 2) In a DDoS attack, the vehicle will send a large number of data packets, and the time consumed by simulating a single scenario will increase sharply, and it will take a lot of time to run all the scenarios. Because a DDoS attack requires multiple vehicles, we set the probability that a vehicle can be controlled by the attacker to be slightly higher in the urban intersection, with three probabilities of 0.3, 0.4, and 0.5, and simulate them separately. During the simulation, normal vehicles broadcast BSM at a frequency of 10 messages per second. Attackers send BSMs from 100 to 500 per second depending on attacking probability. This setting not only guarantees the attack effect but also hides the attacker to a certain extent (because low-frequency messages are more difficult to be detected). Additional configuration parameters can be found in the `omnetpp.ini` file of our public source code (<https://github.com/YangFanAHU/DDoS-attacks-dataset-in-VANETs>).

4.2 Generated Dataset

Our dataset consists of 152 individual simulations, including 2 areas with different vehicle densities and average vehicle speeds, 3 different probabilities of a vehicle being controlled, 5 different kinds of DDoS attacks, and 5 different BSMs sent by attackers. Our dataset is recorded in JSON format like VeReMi. The dataset contains two different forms. First, we record original information in BSMs received by vehicles. It has labels that mark the DDoS type and message tampering type: {*“sendtime”*, *“rcvtime”*, *“senderID”*, *“messageID”*, *“position”*, *“speed”*, *“heading”*, *“ddosType”*, *“msgType”*} The second way is to record some statistics in the time window [8], such as the total BSMs received by vehicle in the time window. It can be regarded as features extracted from the first dataset. The next section will present testing results on some machine learning algorithms using these features. Researchers also can extract other features from original BSMs. We set the time window to 2 s. The full format is presented as follows: {*“received BSMs”*, *“receivedIDs”*, *“Pfid_max”*, *“IAT_mean”*, *“IAT_std”*, *“IAT_max”*, *“IAT_min”*, *“hasAttackers”*} where Pfid is the number of BSMs received from the same id and IAT is the time between two packets received by a vehicle.

We record messages received by each vehicle and statistics within the time window on a vehicle-by-vehicle basis. The dataset has a summary file of all vehicle data to facilitate model training. Besides messages received by vehicles, a groundtruth file is also included in each scenario to record true information about vehicles. All the simulation results can be found in Github (<https://github.com/YangFanAHU/DDoS-attacks-dataset-in-VANETs>).

4.3 Simulation Results

We counted some data generated from simulations to verify the effectiveness of DDoS attacks. Figure 4 shows the total number of BSMs received by each vehicle within 75 s without DDoS attack in the simulation area 3600, 3500–4400, 5000. From 7 to 65336, on average each vehicle receives 25252.71 BSMs (Due to the short driving time of some vehicles in the simulation area, the number of BSMs received by these vehicles is small). As shown in Fig. 5, in the DDoS attack scenario, the average number of BSMs received by each vehicle is 40665.31, which is significantly higher than that in the normal scenario. Figure 6 and Fig. 7 show the number of vehicle packets lost in the two simulated scenarios respectively. We can see that even without DDoS attacks, the vehicular network in high-density areas is very congested, and the average number of packets lost in 75 s is 10356.17. After being attacked by DDoS, the average number of total lost packets is 19543.87. DDoS attack aggravates the congestion level of the vehicular network, and a large portion of the BSMs received by the vehicle is from the attacker.

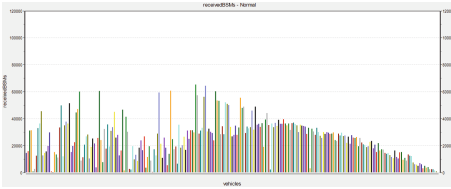


Fig. 4. Total received BSMs in the normal scenario.

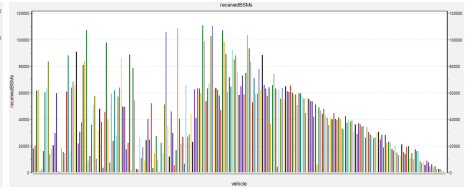


Fig. 5. Total received BSMs in the DDoS scenario.

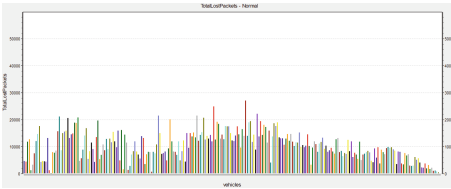


Fig. 6. Total lost packets in the normal scenario.

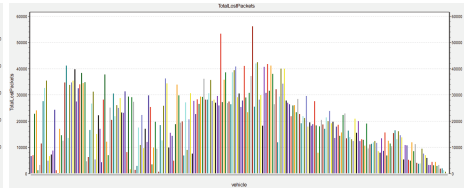


Fig. 7. Total lost packets in the DDoS scenario.

5 Detection

In this section, we test 6 commonly used machine learning algorithms on the dataset presented in this paper, which are Multi-Layer Perceptron (MLP) [20], Random Forest (RF) [3], K-Nearest Neighbor (KNN) [14], Support Vector Classification (SVC) [18], Decision Tree (DT) [19] and Gaussian Naive Bayes (GNB) [16]. All the algorithms are written in Python 3.9. We utilize scikit-learn library

to implement these machine learning models with all parameters default. Both the training set and the test set come from the dataset recorded in the form of time windows, of which the training set contains 12605 pieces of data, and the test set has four parts. 1) Constant DDoS with 4046 pieces of data; 2) Increasing DDoS with 4046 pieces of data; 3) Pulse attack with 4046 pieces of data; 4) Increasing-pulse with 4045 pieces of data. Since the time window dataset is already the feature extracted from the original BSM dataset, we do not process the dataset in this part and use it directly for training. The detection results are presented in the form of a confusion matrix, as shown in Fig. 8 and Fig. 9. From the first to the fourth quadrant of the matrix are False negative (FN),

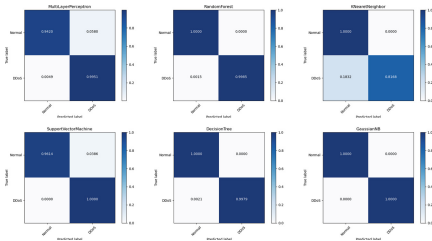


Fig. 8. Confusion matrix - detection results of constant rate DDoS attacks.

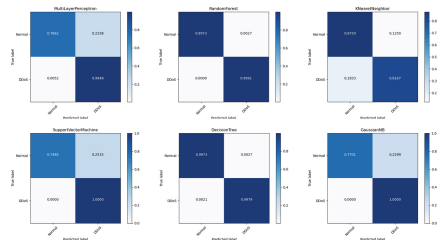


Fig. 9. Confusion matrix - detection results of increasing rate DDoS attacks.

True Negative (TN), False Positive (FP), and True Positive (TP), which represent normal traffic determined as DDoS attack, normal traffic determined as normal traffic, DDoS attack traffic detected as normal traffic and correctly detected DDoS traffic. Using the confusion matrix, we calculated 4 common metrics for evaluating machine learning algorithms, that is *Accuracy*, *Recall*, *Precision*, and *F1score*. Table 2 shows the detection results on the constant DDoS test set. Except KNN, all the detection algorithms perform well, with four indicators above 0.90, especially the accuracy, which is all higher than 0.99. The performance of KNN is very poor, the accuracy is only 0.81, while the recall reaches 0.99, which shows that KNN is seriously misjudging normal traffic. Table 3 presents the detection results of the six algorithms on the Increasing rate attack test set. The overall performance is a little worse than the former because the small-volume attack traffic in the early stage of the attack is similar to the normal traffic, which is difficult to be detected. For Pulse attacks, detection on the current features set almost fails. The best performing DT achieves only 88.03% accuracy. Other algorithms are less than 80% accuracy. SVC is even less than 50%. In fact, this does not mean that the traffic of the pulse attack is indistinguishable from the normal traffic. We observed data packets received by some vehicles and found that in some time windows marked as containing attackers, the current vehicle only received a few BSMs from the attacker (sometimes only one, if the BSM information has not been modified, this is not regarded

Table 2. Detection result of constant rate DDoS attacks

	Detection result			
	Accuracy	Recall	Precision	F1 score
MLP	98.49%	98.63%	99.51%	99.07%
RF	99.87%	100%	99.84%	99.92%
KNN	81.71%	100%	81.67%	89.91%
SVC	99.25%	99.09%	100%	99.54%
DT	99.82%	100%	99.78%	99.89%
GNB	100%	100%	100%	100%

Table 3. Detection result of increasing rate DDoS attacks

	Detection result			
	Accuracy	Recall	Precision	F1 score
MLP	94.09%	93.24%	99.48%	96.26%
RF	99.82%	99.93%	99.84%	99.89%
KNN	81.68%	99.96%	81.67%	89.90%
SVC	93.79%	92.39%	100%	96.04%
DT	99.77%	99.93%	99.78%	99.56%
GNB	94.48%	93.24%	100%	96.50%

as being attacked) due to transmission distance limitations and high packet loss due to congested networks.

6 Conclusion

In this study, we provide a DDoS attack dataset for misbehavior (intrusion) detection in vehicular networks. This dataset contains different types of DDoS attacks combined with different tampered messages that are difficult to detect. We tested some machine learning algorithms on our dataset and presented the detection results. This dataset enables researchers to compare their studies with others and improve the performance of detection algorithms for DDoS attacks on vehicular networks. In future, we plan to design a detection mechanism for DDoS attacks in vehicular networks, particularly for increasing rate and pulse attacks. Moreover, we will make our dataset more realistic by selecting a larger area and increasing the simulation time.

References

1. Alhaidari, F.A., Alrehan, A.M.: A simulation work for generating a novel dataset to detect distributed denial of service attacks on vehicular ad hoc network systems. *Int. J. Distrib. Sens. Netw.* **17**(3), 1–25 (2021)
2. Belenko, V., Krundyshev, V., Kalinin, M.: Synthetic datasets generation for intrusion detection in VANET. In: 11th International Conference on Security of Information and Networks, pp. 1–6. ACM (2018)
3. Belgiu, M., Drăguț, L.: Random forest in remote sensing: a review of applications and future directions. *ISPRS J. Photogramm. Remote. Sens.* **114**, 24–31 (2016)
4. Codeca, L., Frank, R., Engel, T.: Luxembourg sumo traffic (lust) scenario: 24 hours of mobility for vehicular networking research. In: Vehicular Networking Conference (VNC), pp. 1–8. IEEE (2015)
5. Gonçalves, F., et al.: Synthesizing datasets with security threats for vehicular ad-hoc networks. In: Global Communications Conference (GLOBECOM), pp. 1–6. IEEE (2020)
6. Haidar, F., Kamel, J., Jemaa, I.B., Kaiser, A., Lonc, B., Urien, P.: Dare: a reports dataset for global misbehavior authority evaluation in c-its. In: 91st Vehicular Technology Conference (VTC 2020-Spring), pp. 1–6. IEEE (2020)

7. van der Heijden, R.W., Lukaseder, T., Kargl, F.: VeReMi: a dataset for comparable evaluation of misbehavior detection in VANETs. In: Beyah, R., Chang, B., Li, Y., Zhu, S. (eds.) *SecureComm 2018*. LNICST, vol. 254, pp. 318–337. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01701-9_18
8. Idhammad, M., Afdel, K., Belouch, M.: Semi-supervised machine learning approach for DDoS detection. *Appl. Intell.* **48**(10), 3193–3208 (2018)
9. Kadam, N., Krovi, R.S.: Machine learning approach of hybrid KSVN algorithm to detect DDoS attack in VANET. *Int. J. Adv. Comput. Sci. Appl.* **12**(7), 718–722 (2021)
10. Kamel, J., Ansari, M.R., Petit, J., Kaiser, A., Jemaa, I.B., Urien, P.: Simulation framework for misbehavior detection in vehicular networks. *IEEE Trans. Veh. Technol.* **69**(6), 6631–6643 (2020)
11. Kamel, J., Wolf, M., van der Heijder, R.W., Kaiser, A., Urien, P., Kargl, F.: VeReMi extension: a dataset for comparable evaluation of misbehavior detection in VANETs. In: *International Conference on Communications (ICC)*, pp. 1–6. IEEE (2020)
12. Kolandaisamy, R., Noor, R.M., Z'aba, M.R., Ahmady, I., Kolandaisamy, I.: Adapted stream region for packet marking based on DDoS attack detection in vehicular ad hoc networks. *J. Supercomput.* **76**(8), 5948–5970 (2020)
13. Krajzewicz, D., Erdmann, J., Behrisch, M., Bieker, L.: Recent development and applications of sumo-simulation of urban mobility. *Int. J. Adv. Syst. Meas.* **5**(3&4) (2012)
14. Liao, Y., Vemuri, V.R.: Use of k-nearest neighbor classifier for intrusion detection. *Comput. Secur.* **21**(5), 439–448 (2002)
15. Mirkovic, J., Reiher, P.: A taxonomy of DDoS attack and DDoS defense mechanisms. *ACM SIGCOMM Comput. Commun. Rev.* **34**(2), 39–53 (2004)
16. Mukherjee, S., Sharma, N.: Intrusion detection using Naive Bayes classifier with feature reduction. *Procedia Technol.* **4**, 119–128 (2012)
17. Poongodi, M., Hamdi, M., Sharma, A., Ma, M., Singh, P.K.: DDoS detection mechanism using trust-based evaluation system in VANET. *Access* **7**, 183532–183544 (2019)
18. Pradhan, A.: Support vector machine-a survey. *Int. J. Emerg. Technol. Adv. Eng.* **2**(8), 82–85 (2012)
19. Safavian, S.R., Landgrebe, D.: A survey of decision tree classifier methodology. *IEEE Trans. Syst. Man Cybern.* **21**(3), 660–674 (1991)
20. Sarle, W.S.: Neural networks and statistical models. In: *19th Annual SAS Users Group International Conference*, pp. 1–13. SAS (1994)
21. Sommer, C., German, R., Dressler, F.: Bidirectionally coupled network and road traffic simulation for improved IVC analysis. *IEEE Trans. Mob. Comput. (TMC)* **10**(1), 3–15 (2010)
22. Varga, A.: Discrete event simulation system. In: *European Simulation Multiconference (ESM 2001)*, pp. 1–7. EUROSIS (2001)



Vehicle-Road Cooperative Task Offloading with Task Migration in MEC-Enabled IoV

Jiarong Du, Liang Wang^(✉), Yaguang Lin, and Pengcheng Qian

School of Computer Science, Shaanxi Normal University, Shaanxi, China
{dujiarong,wangliang,light,qianpengcheng}@snmu.edu.cn

Abstract. Mobile edge computing (MEC) is considered as a key technology for addressing computation-intensive and delay-critical applications in the Internet of vehicles (IoV). In MEC-enabled IoV, vehicles lighten their computing load by offloading tasks to edge servers. However, the high speed mobility of vehicles and time-varying network environment brings tough challenges to task offloading. In addition, considering only roadside units (RSUs) or vehicles as offloading objects lead to the waste of computing resources and increase the process delay of task. To this end, we formulate the reduction of task processing delay and improvement of service reliability as an utility maximization problem and propose a distributed vehicle-road cooperative task offloading scheme with task migration. Then we use RSUs and surrounding vehicles as offloading objects and divide offloading tasks into multiple subtasks for offloading objects and local parallel processing, which improves the utilization rate of computing resources. Meanwhile, we reduce the task processing failure by migrating the computing results of offloading subtasks. The offloading scheme is formulated as a mixed-integer nonlinear optimization problem, and a multi-agent deep Q-learning network (MADQN) algorithm is proposed to find the near-optimal offloading objects and number of offloading subtasks. Simulation results show that the proposed approach significantly improves the total task processing speed and service reliability.

Keywords: Mobile edge computing · Internet of vehicles · Task offloading · Multi-agent deep Q-learning network

1 Introduction

Thanks to the development of mobile communication technology, vehicles as the new mobile device, can realize information sharing among vehicles, users and infrastructure, thus the concept of Internet of vehicles (IoV) came into

This work was supported in part by the National Natural Science Foundation of China under Grant 62071283, Grant 62102240, and Grant 61872228, in part by the China Postdoctoral Science Foundation under Grant 2020M683421, in part by the Key R&D Program of Shaanxi Province under Grant 2020ZDLGY10-05.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022
L. Wang et al. (Eds.): WASA 2022, LNCS 13473, pp. 261–272, 2022.
https://doi.org/10.1007/978-3-031-19211-1_22

being [1]. The IoV plays an important role in intelligent transportation systems (ITSs) [2] and novel smart cyber-physical systems (CPSs) [3], like data sharing [4] and privacy protection [5]. With the further development of IoV, mobile devices generate huge amounts of data [6], a variety of computation-intensive on-board applications have emerged. Autonomous driving applications, for example, generate a lot of perceptual data [7, 8]. The vehicle's computing and storage resources are usually limited to serve such applications, so how to process such applications in the IoV is a serious challenge [9].

Fortunately, the emergence of mobile edge computing (MEC) offers a promising approach to tackle such issues. MEC is a new computing architecture in which computing, storage and communication services are deployed at the edge of networks closer to data sources [10]. In MEC-enabled IoV (MEC-IoV), vehicles utilize edge objects (ENs) resources through task offloading, which significantly reduces task processing delay. However, task offloading also brings new challenges. The most obvious is the additional cost of transmission. What's worse, the vehicles may not receive the computing results due to the fast movement of vehicles and the limited communication range of edge objects [11], which reduces the service reliability. Therefore, in order to meet the needs of computation-intensive applications in time-varying IoV, an efficient offloading scheme of tasks is worth studying.

Existing works are mainly divided into the vehicle-to-vehicle (V2V) and the vehicle-to-road infrastructure (V2R) offloading scheme. The V2V offloading schemes usually only consider the computing resources of vehicles on the road and regard them as vehicle edge node (VEN). The schemes that considers only the task offloading between vehicles is presented in [12, 13]. Buda et al. [14] proposes a scheme using vehicle clusters as edge server to reduce processing delay. A WiFi-assisted offloading method in connected vehicle scenarios is proposed in [15] to maximize the overall transmission efficiency. Since RSUs have more computing resources than vehicles, V2R scenarios generally perform better. Notably, most V2R offloading schemes are designed to minimize task processing delays [16, 17]. However, focusing only on reducing the offloading delay and offloading almost all tasks to the RSU, the vehicle may not be able to receive the computing results in the case of high speed movement, resulting in offloading failure. So we attempt to combine V2V with V2R and propose a distributed vehicle-road cooperative offloading scheme with task migration. Due to the above problem is a mixed-integer nonlinear optimization problem, traditional optimization methods are difficult to solve it. We use multi-agent deep Q-learning network (MADQN) algorithm in Multi-agent deep reinforcement learning (MADRL) to solve the above problem. The main contributions of our work are summarized as follows:

- In the time-varying MEC-IoV, we integrate RSUs and vehicles computing resources and propose a distributed vehicle-road cooperative offloading scheme. This offloading scheme divides the offloading task into several sub-tasks that can be processed distributed on ENs. ENs includes RSUs and VENs.

- In order to improve service reliability, we added task migration mechanism in the above offloading scheme. Task migration overcomes the shortcoming of limited communication range of ENs in the MEC-IoV.
- We formulate the problem as an MDP and propose two MADQN algorithms to find the appropriate offloading object and the number of offloading sub-tasks to achieve the approximate optimal offloading performance. Simulation results show that the proposed scheme significantly improves the total task processing speed and service reliability.

2 System Model

This section describes the system architecture for task offloading in MEC-IoV. In this architecture, we focus on a multi-vehicle and timeslot-based scenario, and the total time slot is T . The vehicle that generates the offloading task is defined as the offloading vehicle (OV). As shown in Fig. 1, a stretch of the highway is cooperatively covered by a row of RSUs located at one side of the road. The RSUs are connected through fibers. RSUs and vehicles can communicate with each other via wireless links within its communication range.

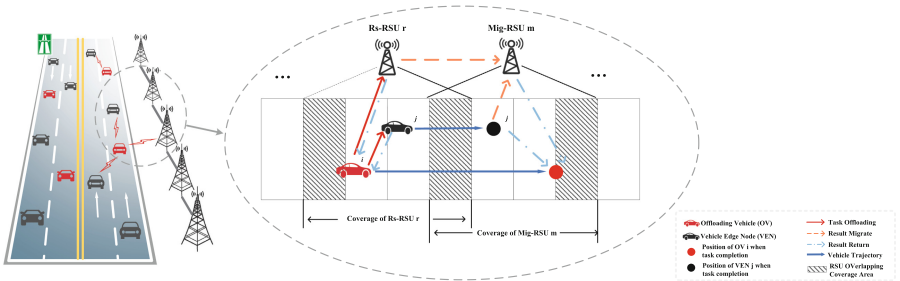


Fig. 1. Highway task offloading scenario.

The set of RSUs is denoted by $\mathcal{R} = \{1, \dots, R\}$, where R is the number of RSUs. VENs is denoted as $\mathcal{V} = \{1, \dots, V\}$, and OVs is denoted as $\mathcal{O} = \{1, \dots, N\}$, V and N are the number of VENs and OVs, respectively. OV i ($i \in \mathcal{O}$) has the option of offloading tasks to RSU or VEN. The RSU that receives the offloading tasks is called the receiver RSU (Rs-RSU) and is denoted as r ($r \in \mathcal{R}$). The VEN receives offloading tasks is denoted as j ($j \in \mathcal{V}$). Note that OVs select the nearest EN to offload tasks. We use a binary variable $\alpha_i(t)$ to indicate OV i offload task to Rs-RSU or VEN in time slot t , $\alpha_i(t) = 0$ indicates offloading to Rs-RSU, and $\alpha_i(t) = 1$ indicates offloading to VEN. In particular, in order to reduce the probability of task offload failure, we specify that the offload object can migrate the computing result of the task to the next RSU, called migrator RSU (Mig-RSU), denoted as m ($m \in \mathcal{R}$).

OV i generates an offloading task in time slot t . Assume that all tasks can be completed before the next time slot. The offloading task that generated by the OV i in time slot t is denoted as $H_i(t) = \{W_i(t), Z_i(t)\} (t \in T)$, which $W_i(t)$ represents the amount of computation tasks, $Z_i(t)$ (in CPU cycles per byte) indicates the number of computation cycles needed to execute 1 bit of data for task $H_i(t)$. $H_i(t)$ can be divided into several subtasks of equal size and each subtask can be processed in any order. The number of subtasks is a constant, denoted as $X (X \in N^*)$. The number of subtasks of vehicle i offloaded to the EN in time slot t is denoted as $x_i(t)$ ($0 \leq x_i(t) \leq X$), where $x_i(t)$ is an integer variable.

2.1 Communication Model

The communication model follows the reference [18]. Therefore, there are two communication scenarios: vehicle to RSU (V2R) and vehicle to vehicle (V2V). In V2R scenario, the offloading transmission rate for OV i to Rs-RSU r is

$$r_{ir}(t) = B_r \log_2 \left(1 + \frac{P_r \cdot 10^{-PL_r(d_{ir}(t))/10}}{N} \right), \quad (1)$$

where $PL_r(d_{ir}(t))$ is the pathloss between vehicle and RSU which is defined as $PL_r(d_{ir}(t)) = 128.1 + 37.6 \lg d_{ir}(t)$, $d_{ir}(t)$ represents the distance between the OV i and the Rs-RSU r in kilometres in time slot t . P_r and B_r represent the transmit power and bandwidth for OV to Rs-RSU respectively. N is denoted as the power of the Gaussian noise. In V2V scenario, the offloading transmission rate for OV i to VEN j is

$$r_{ij}(t) = B_j \log_2 \left(1 + \frac{P_j \cdot 10^{-PL_j(d_{ij}(t))/10}}{N} \right), \quad (2)$$

where $PL_j(d_{ij}(t))$ represents the distance between the OV i and the VEN j , which calculation method is referenced in [18], and $d_{ij}(t)$ represents the distance between the OV i and the VEN j in kilometres in time slot t . P_j and B_j represents the transmit power and the bandwidth for OV i to VEN j respectively. Thus, the transmission delay for OV i to offloading object is

$$T_{io}(t) = (1 - \alpha_i(t)) \cdot \frac{x_i(t)W_i(t)}{Xr_{ir}(t)} + \alpha_i(t) \cdot \frac{x_i(t)W_i(t)}{Xr_{ij}(t)}. \quad (3)$$

2.2 Computation Model

The offloading subtasks processing time of OV i shown is

$$T_{io}^P(t) = (1 - \alpha_i(t)) \cdot \frac{Z_i(t)x_i(t)W_i(t)}{XC_{ir}(t)} + \alpha_i(t) \cdot \frac{Z_i(t)x_i(t)W_i(t)}{XC_{ij}(t)}, \quad (4)$$

where $C_{ir}(t)$ and $C_{ij}(t)$ represent the computing resources allocated to OV i by Rs-RSU r and VEN j in time slot t , respectively. And the processing time of locally executed task can be computed as

$$T_{il}^P(t) = \frac{(X - x_i(t))Z_i(t)W_i(t)}{XC_i(t)}, \tag{5}$$

where $C_i(t)$ denotes the computing capability of OV i . Due to the time variability of the IoV, we assume that the computing resources obtained by OVs from ENs in each time slot are random. Local processing is the same. Thus, the total processing time of OV i can be formulated as follows:

$$T_i^C(t) = \max\{T_{io}(t) + T_{io}^P(t), T_{il}^P(t)\}. \tag{6}$$

3 Problem Formulation

We aim to minimize the OVs' total processing delay and improve service reliability for OVs. To evaluate service reliability, we rule task processing failure if the result returns failure or the total processing delay of the task exceeded the maximum tolerated delay T_{max} . We introduce a binary variable $F_i(t)$ to indicate whether OV i has failed to processed or not, $F_i(t) = 1$ represents task processing succeeded, $F_i(t) = 0$ represents processing failed. Then, in order to reduce the probability of offloading failure, we introduce task migration mechanism. The proposed offloading scheme is described in detail in Algorithm 1. Thus, we model the above problem as a utility maximization problem and denote it as:

$$\max_{x,\alpha} \sum_{t=1}^T \left(\frac{W_i(t)}{T_i^C(t)} \cdot F_i(t) + \gamma W_i(t) \cdot (1 - F_i(t)) \right), \tag{7}$$

- s.t. C1: $\alpha = \{0, 1\}$, C2: $n \leq 1$, C3: $T^C(t) \leq T_{max}$,
- C4: $\sum_{i=1}^N \alpha_i(t)C_{ir}(t) \leq C_r$, C5: $\sum_{i=1}^N (1 - \alpha_i(t))C_{ij}(t) \leq C_v$,
- C6: $C_i(t) \leq C_v$, C7: $N + V = V_{total}$,

where γ is the penalty factor for the case when the computing service fails. V_{total} is the total number of vehicles. In particular, we use task processing speed to refer to the total processing delay. In the set of constraints, constraint C1 guarantees OV i must select RSU or VEN as offload object. Constraint C2 make sure that the result migration can occur at most once. Constraint C3 ensures that the total processing delay must be less than the maximum tolerated delay, or the offloading fails. Constraints C4, C5 and C6 ensure that the sum of computing resources obtained by all OVs in time slot t is less than the computing resources provided by RSU and vehicle. Constraint C7 ensures that the sum of OVs and VENs remains the same.

Algorithm 1. Distributed vehicle-road cooperative offloading scheme with task migration

```

1: Get  $\alpha_i(t), x_i(t)$  by MADQN
2: Calculate the transmission delay  $T_{io}(t)$  in (3), the offloading subtasks processing
   time  $T_{io}^P(t)$  in (4) and the total processing time  $T_i^C(t)$  in (6)
3: if  $T_i^C(t) < T_{max}$  then
4:   if  $\alpha_i(t) == 0$  then
5:     if Task  $H_i(t)$  processing is complete and the OV  $i$  is still within Rs-RSU  $r$ 
       communication range then
6:       No task migration required and  $F_i(t) = 1$ 
7:     else
8:       The processing results of the task to be migrated to the next Mig-RSU  $m$ 
9:       if OV  $i$  is within communication range of the Mig-RSU  $m$  then
10:         $F_i(t) = 1$ 
11:      else
12:         $F_i(t) = 0$ 
13:      end if
14:    end if
15:   else
16:     if Task  $H_i(t)$  processing is complete and the OV  $i$  is still within VEN  $j$ 
       communication range then
17:       No task migration required and  $F_i(t) = 1$ 
18:     else
19:       The processing results of the task need to be migrated from VEN  $j$  to the
       nearest Mig-RSU  $m$ 
20:       if OV  $i$  is within communication range of the Mig-RSU  $m$  then
21:         $F_i(t) = 1$ 
22:      else
23:         $F_i(t) = 0$ 
24:      end if
25:    end if
26:   end if
27: else
28:    $F_i(t) = 0$ 
29: end if

```

4 MADQN-Based Offloading Strategy

In order to solve the above problems effectively, we adopt Deep Q-learning Network (DQN) [19] to find the optimal solution with large state-space, because the traditional dynamic optimization problem is difficult to deal with the problem with unknown state transition probability and the large state-action space. We propose two MADQN algorithms, namely self-interested DQN and cooperative DQN to solve the multi-vehicle task offloading problem.

We consider a timeslot-based process and assume the driving behavior of OVs can be considered as a Markov decision process (MDP) with finite horizon, which is defined by a tuple $(\mathbb{S}, \mathbb{A}, \mathbb{T}, \mathbb{R})$. \mathbb{S} represents the set of possible

system states, the state for OV i in time slot t can be calculated as $S_i(t) = \{C_{ir}(t), C_{ij}(t), r_{ir}(t), r_{ij}(t), C_i(t), W_i(t), H_i(t), X\}$. \mathbb{A} represents the set of actions, the action taken by the OV i in time slot t is expressed as $a_i(t) = \{\alpha_i(t), x_i(t)\}$. $\mathbb{T} = \{p(S_{t+1}|S_t, a_t)\}$ is the set of transition probabilities, and \mathbb{R} is the reward function. To maximize total utility, we define the reward function as $R_i(s_{it}, a_{it}) = (W_i(t)/T_i^C(t)) \cdot F_i(t) + \gamma W_i(t) \cdot (1 - F_i(t))$.

Algorithm 2. Multi-agent deep Q-learning network

```

1: Initialize DQN-networks for all OVs randomly
2: for each episode do
3:   Update OVs and VENs location
4:   for each time slot  $t$  do
5:     for each OV  $i$  do
6:       Observe  $S_i(t)$ 
7:       Choose action  $a_i(t)$  with  $\varepsilon$ -greedy policy
8:       Perform action  $a_i(t)$  to observe reward  $R_i(t)$  and next state  $S_i(t+1)$ 
9:       Updated VENs and RSUs computing resources
10:    end for
11:    for each OV  $i$  do
12:      if Self-interested MADQN then
13:        Store  $(S_i(t), a_i(t), R_i(t), S_i(t+1))$  in experience replay pool  $D_i$ 
14:      else
15:        Store  $(S_i(t), a_i(t), \sum_{i=1}^N R_i(t), S_i(t+1))$  in experience replay pool
16:      end if
17:    end for
18:    for each OV  $i$  do
19:      Sample random minibatch from  $D_i$ 
20:      Perform a gradient descent step on (8) to optimize error between Q-network
      and learning targets
21:    end for
22:  end for
23: end for

```

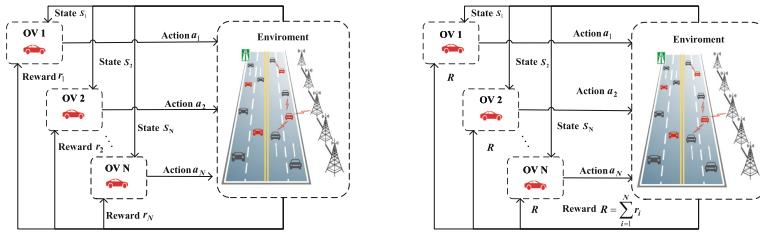
4.1 DQN Algorithm

In the IoV, due to the large number of environment states, we have to employ a deep neural network (DNN) to approximate the Q-function. In order to approximate the Q-table, we need to train the neural network by sample, which is selected from the experience replay pool and θ that is the set of weight parameters of every layer of the network. The value function can be expressed as $Q(s_t, a_t, \theta)$. The storage format of data in the experience replay pool is (s_t, a_t, R_t, s_{t+1}) , which contain the current state, current action, current reward and next state. Each time updating parameters, a batch of stored experience chosen randomly from the replay memory is used as samples to train the network. To update the network, by taking minimum mean square error, we define a loss function as

$$L_t(\theta_t) = \mathbb{E}[(Q_t^{target} - Q(s_t, a_t|\theta))^2], \quad (8)$$

where Q_t^{target} is the target value that represents the optimization object of the predict network. Nevertheless, if we use the same DNN to obtain the target value, the optimization object will be changed with the parameter θ at each iteration. Therefore, we apply the target network which has the same structure with the predict network, except that the parameter update of the target network θ^- is t_{copy} time slots later than that of the predict network. So we can calculate the target value $Q_t^{target} = \mathbb{E}[R_t + \lambda Q(s_t, \arg \max_{a_{t+1}} Q(s_{t+1}, a_{t+1} | \theta^-))]$.

For each state, the algorithm selects an action through the predict network. The predict network evaluates the value after each possible action, then the best action based on predicting result will be taken with possibility $1 - \varepsilon$.



(a) Self-interested DQN.

(b) Cooperation DQN.

Fig. 2. The framework of two MADQN algorithms.

4.2 MADQN-Based Solution

In MADQN, each OV learns strategies based on its own actions and treats other vehicles as part of the environment [20]. The framework of the two algorithms is shown in Fig. 2. The difference between the two MADQN algorithms lies in the calculation design of the reward function:

- Self-interested MADQN: Each OV is selfish, that is, they independently maximizes its own reward function. For this reason, the reward function in time slot t shown is $R_i(t) = W_i(t)/T_i^C(t) \cdot F_i(t) + \gamma W_i(t) \cdot (1 - F_i(t))$.
- Cooperation MADQN: The objective of each OV is to maximize the sum of the reward of all OVs in slot t , all OV’s have the same reward, expressed as

$$R_i(t) = \sum_{i=1}^N \left\{ W_i(t)/T_i^C(t) \cdot F_i(t) + \gamma W_i(t) \cdot (1 - F_i(t)) \right\}.$$

The details of MADQN are summarized in Algorithm 2.

5 Simulation Results

In this section, we make a numerical simulation to evaluate the performance of our proposed scheme. We consider a 800 m long freeway with 5 RSUs deployed

along by the one side of the road. The distance between RSUs is 150 m and the coverage radius of RSU is 100 m. The OVs has a average speed of 108 km/h and generates tasks every 10 milliseconds. The amount of computation tasks in each time slot follow the uniform distribution on [2, 8] KB. The number of computing cycles required for the OVs to execute 1 bit data for the task is also random in each time slot, ranging from 5 to 10. The VENs travels at a average speed of 90 km/h. Other key parameters are summarized in Table 1.

Table 1. The simulation parameters.

Parameters	Definition	Typical Values
B_r, B_j	The bandwidth reserved for vehicle i to RSU r and VEN j	1, 0.5 MHz
P_r, P_j	The transmit power for vehicle i to RSU r and VEN j	20, 14 dBm
f	The carrier frequency	2 GHz
N	The power of the Gaussian noise	-100 dBm
C_r, C_v	The computing capability of RSU r and vehicle j	80, 120 MHz
R_r, R_j	The communication radius of RSU r and VEN j	100, 50 m
V_{total}	The total number of vehicles on the road	30
D	The size of the replay memory	20000
E	The training episodes	500
α	The learning rate	0.01

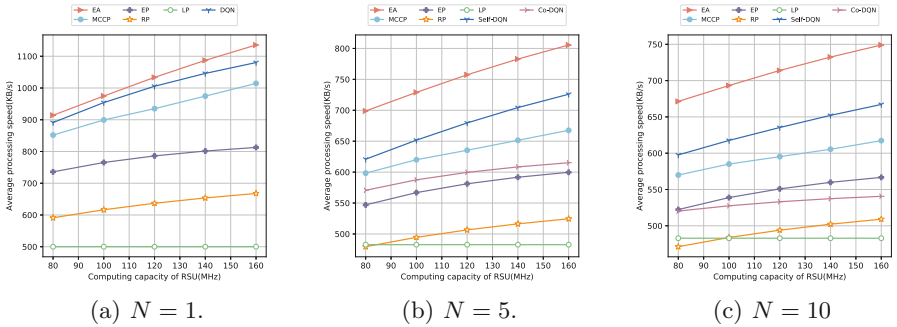


Fig. 3. The influence of RSU computing capacity on the OVs' average processing speed.

In order to verify the superior performance of our proposed scheme under different offloading densities, we first get the optimal strategy by exhaustive algorithm (EA), and then compare the performance of the proposed scheme with the local processed (LP) scheme, random processed (RP), equal processed (EP) scheme and maximum computing capability processed (MCCP) scheme on average offloading speed and task processing failure probability when $N = 1$, $N = 5$ and $N = 10$ respectively. In LP scheme, tasks are processed locally only. In the RP scheme, the selection of offloading object and number of offloading

subtasks is random. In EP scheme, offloading object are randomly selected, and the number of uninstalled subtasks is the same as that of the remaining subtasks. The MCCP scheme requires OVs select the EN with the strongest computing capacity within its communication range as the offloading object. The number of offloading subtasks is determined based on the ratio of computing resources obtained by the OV from the offloading object and the local. Note that in all offloading schemes, OVs are allocated resources in fixed order.

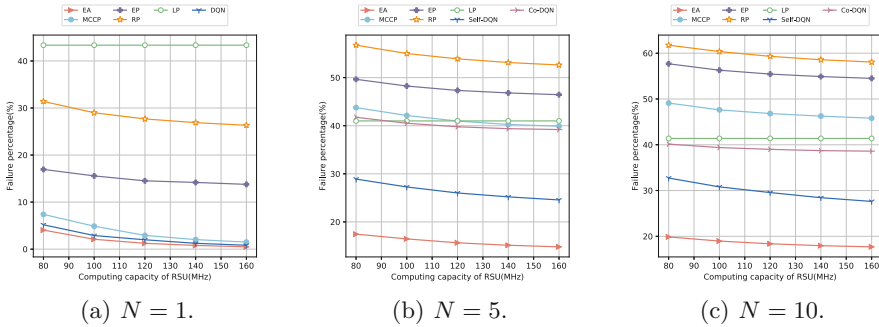


Fig. 4. The influence of RSU computing capacity on the OVs’ failure percentage.

Figure 3 and Fig. 4 describe the task offloading performance of the offloading schemes when the computing capacity of RSU from 80 MHz to 120 MHz. As shown in Fig. 3, with the increasing computing capacity of the RSUs, the average processing speed of offloading schemes increases. This is because when the computing capacity gap between RSUs and vehicles is getting wider, OVs is more inclined to offload tasks to RSUs, which reduces the task processing delay. More importantly, the communication range of RSU is much higher than that of vehicle. Compared with offloading task to VENs, the probability of task offloading failure due to the limited communication range is significantly reduced when the task is offloaded to RSU. Therefore, it can be seen from Fig. 4 that the processing failure percentage of the offloading schemes decreases with the increase of RSU computing capacity.

Figure 5 illustrates how the tasks size influences the offloading performance of the offloading schemes when the tasks size from [4, 8] KB. It can be seen in the figure that when the tasks size increases, the offloading failure percentage of the offloading schemes will be improved.

As can be seen from the above simulation results, the performance of self-interested DQN is significantly better than that of cooperative DQN, because in the case of given computing resources, OVs does not need to maximize the overall cumulative reward through the balance of resource allocation. In this case, maximizing each OV’s own reward is equivalent to maximizing the overall reward. In addition, the self-interested DQN algorithm shows excellent performance under different offloading densities, which further proves that our proposed offloading scheme has strong generalization performance.

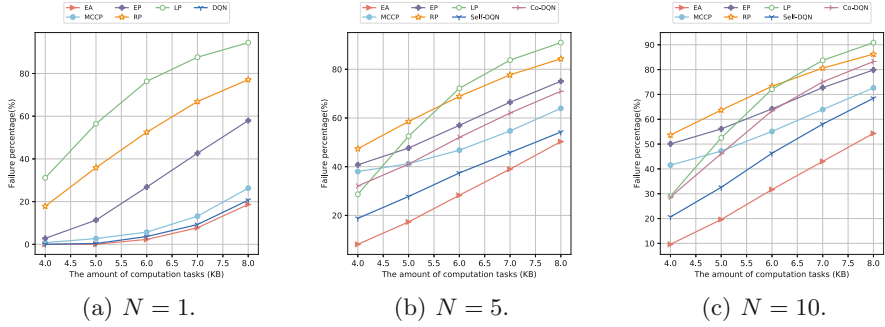


Fig. 5. The influence of tasks size on the OVs' processing failure percentage.

6 Conclusions

In this paper, we propose a distributed vehicle-road cooperative offloading scheme with task migration in order to reduce total processing time and improve service reliability. The problem is formulated as a mixed-integer nonlinear optimization problem, and we use the MADQN algorithm to solve the problem. Simulation results verify the effectiveness of the proposed scheme. In the future, we will take into account more complex vehicle driving scenario and the needs of different on-board applications. In addition, random partitioning of tasks and subtasks dependency will also be considered in order to be more in line with the actual application scenarios.

References

1. Contreras-Castillo, J., Zeadally, S., Guerrero-Ibañez, J.A.: Internet of vehicles: architecture, protocols, and security. *IEEE Internet Things J.* **5**(5), 3701–3709 (2018)
2. Xu, H., Cai, Z., Li, R., Li, W.: Efficient CityCam-to-edge cooperative learning for vehicle counting in ITS. *IEEE Trans. Intell. Transp. Syst.* 1–12 (2022). <https://doi.org/10.1109/TITS.2022.3149657>
3. Cai, Z., Zheng, X.: A private and efficient mechanism for data uploading in smart cyber-physical systems. *IEEE Trans. Netw. Sci. Eng.* **7**(2), 766–775 (2020)
4. Zheng, X., Cai, Z.: Privacy-preserved data sharing towards multiple parties in industrial IoTs. *IEEE J. Sel. Areas Commun.* **38**(5), 968–979 (2020)
5. Cai, Z., He, Z.: Trading private range counting over big IoT data. In: 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), pp. 144–153 (2019). <https://doi.org/10.1109/ICDCS.2019.00023>
6. Cai, Z., Zheng, X., Wang, J., He, Z.: Private data trading towards range counting queries in internet of things. *IEEE Trans. Mob. Comput.* (2022)
7. Xiong, Z., Cai, Z., Han, Q., Alrawais, A., Li, W.: ADGAN: protect your location privacy in camera data of auto-driving vehicles. *IEEE Trans. Industr. Inf.* **17**(9), 6200–6210 (2020)

8. Cai, Z., Duan, Z., Li, W.: Exploiting multi-dimensional task diversity in distributed auctions for mobile CrowdsMensing. *IEEE Trans. Mob. Comput.* **20**(8), 2576–2591 (2020)
9. Peng, H., Liang, L., Shen, X., Li, G.Y.: Vehicular communications: a network layer perspective. *IEEE Trans. Veh. Technol.* **68**(2), 1064–1078 (2019)
10. Lin, Y., Wang, X., Ma, H., Wang, L., Hao, F., Cai, Z.: An efficient approach to sharing edge knowledge in 5G-enabled industrial internet of things. *IEEE Trans. Industr. Inform.* (2022)
11. Wang, L., Ye, H., Liang, L., Li, G.Y.: Learn to compress CSI and allocate resources in vehicular networks. *IEEE Trans. Commun.* **68**(6), 3640–3653 (2020)
12. Chen, C., Zhang, Y., Wang, Z., Wan, S., Pei, Q.: Distributed computation offloading method based on deep reinforcement learning in ICV. *Appl. Soft Comput.* **103**, 107108 (2021)
13. Sun, Y., et al.: Adaptive learning-based task offloading for vehicular edge computing systems. *IEEE Trans. Veh. Technol.* **68**(4), 3061–3074 (2019)
14. Buda, S., Guleng, S., Wu, C., Zhang, J., Yau, K.L.A., Ji, Y.: Collaborative vehicular edge computing towards greener ITS. *IEEE Access* **8**, 63935–63944 (2020)
15. Qin, H., Cao, B., Chen, W., Xiao, X., Peng, Y.: WiFi-assisted control traffic offloading in vehicular environments. *IEEE Trans. Veh. Technol.* **69**(4), 4542–4547 (2020)
16. Luo, Q., Li, C., Luan, T.H., Shi, W., Wu, W.: Self-learning based computation offloading for internet of vehicles: model and algorithm. *IEEE Trans. Wireless Commun.* **20**(9), 5913–5925 (2021)
17. Wang, T., Luo, X., Zhao, W.: Improving the performance of tasks offloading for internet of vehicles via deep reinforcement learning methods. *IET Commun.* (2022)
18. Winner, I.: WINNER II channel models. IST-4-027756 WINNER II, D. 1. 1. 2 V1.2 (2007)
19. Mnih, V., et al.: Playing atari with deep reinforcement learning. *Computer Science* (2013)
20. Liang, L., Ye, H., Li, G.Y.: Spectrum sharing in vehicular networks based on multi-agent reinforcement learning. *IEEE J. Sel. Areas Commun.* **37**(10), 2282–2292 (2019)



Freshness-Aware High Definition Map Caching with Distributed MAMAB in Internet of Vehicles

Qixia Hao, Jiaxin Zeng, Xiaobo Zhou^(✉), and Tie Qiu

College of Intelligence and Computing, Tianjin University, Tianjin 300350, China
{qxhao, jiaxinzeng, xiaobo.zhou, qiutie}@tju.edu.cn

Abstract. The high-definition (HD) map is the foundation for autonomous driving, which has a huge data volume and needs to be updated frequently. To ensure low download latency, HD map contents are usually pre-cached at roadside units (RSU) or vehicles. However, the HD map contains a lot of dynamic data, and maintaining its freshness is crucial for ensuring driving safety, which is ignored by the existing HD map caching methods. In this paper, we propose a freshness-aware HD map caching method to minimize both download latency and loss of freshness. First, we introduce a cost function to incorporate both the download latency and the loss of freshness. Next, we formulate the HD map caching problem as an optimization problem to minimize the total cost. To reduce computation complexity, we decompose the original problem into two subproblems. Consequently, we propose a freshness-aware vehicle request algorithm to optimize vehicle request decisions and then leverage a distributed multi-agent multi-armed bandit (MAMAB) algorithm to make optimal caching decisions. Finally, simulation results verify that the proposed freshness-aware HD map caching method outperforms other baseline methods.

Keywords: High-definition map · Edge caching · Freshness · Internet of vehicle · Multi-agent multi-armed bandit

1 Introduction

Autonomous driving is an important application to the Internet of Vehicles (IoV). In order to meet safety and accuracy, high-definition (HD) maps with precise and rich semantic information about roads are important for correct path planning and driving decision-making [1]. HD maps have a huge data volume and contain many dynamic data. For instance, the amount of HD map data used by Google reaches 1 GB/mile [2]. Therefore, it is infeasible to pre-cache all HD maps on the vehicles. Furthermore, vehicles in the same area may request the same HD maps, which imposes tremendous pressure on the backhaul link, resulting in unacceptable latency.

In order to reduce the download latency, the HD maps can be pre-cached in the roadside unit (RSU) by using the edge caching technology [3]. If the

content popularity within the serving area of RSUs is known to follow a particular distribution, such as Zipf distribution [4], we can cache the contents with high popularity at RSUs to maximize the cache hit rate and reduce download latency [5]. In [6], the authors predicted content popularity based on historical vehicle requests through the long short-term memory (LSTM) algorithm. Due to the high mobility of vehicles, the popularity of content changes frequently and is usually difficult to be accurately predicted. Therefore, Yu *et al.* [7] proposed to predict vehicle requests using federated learning, based on which the caching decision is made. Zhao *et al.* [8] proposed to estimate the future connected RSUs using data traces based on vehicle mobility. The authors in [9] considered the dynamic of HD maps and proposed a distributed algorithm to optimize the cache strategy of RSUs. However, due to the limited cache capacity of RSU, it cannot cache all the requested contents.

Inspired by the increasing computation and storage resources of vehicles, some researchers proposed to cache HD maps in both RSUs and vehicles. In this case, the vehicles can fetch HD maps via vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication to further reduce download latency and alleviate the burden on the backhaul link [10]. The authors in [11] proposed a method to first decide the request decisions through a matching method and then based on the request decisions to determine cache placement to minimize the average download latency. Liu *et al.* [12] proposed to form a platoon of vehicles requesting the same HD maps, and Wu *et al.* [13] designed a cluster-based strategy, where vehicles can fetch HD maps via V2V or V2I. In [14], the authors proposed a method in that vehicles can fetch HD maps from their opposite vehicles. However, the HD map contains a lot of dynamic data, such as the position, direction, and speed of pedestrians and vehicles, which is crucial for real-time driving decisions. Additionally, the freshness of dynamic HD maps plays an important role in making the driving decisions, but dynamic HD maps cached in vehicles suffer a loss of freshness compared to those cached in RSUs and the cloud, which is ignored by the previous methods. It may lead to reduced driving safety.

To address these problems, in this paper, we propose an efficient HD map caching strategy to achieve low latency while ensuring the freshness of HD maps. The main contributions of this article are summarized as follows.

- We present a freshness-aware HD map caching method to minimize both download latency and loss of freshness. We introduce a cost function to balance download latency and loss of freshness. The HD map caching problem is then formulated as an optimization problem to minimize the total cost.
- We decompose the formulated problem into two subproblems: a vehicle request problem and a caching placement problem. Then, we design a freshness-aware vehicle request algorithm to make optimal vehicle request decisions. Next, we propose a distributed caching approach based on a multi-agent multi-armed bandit (MAMAB) algorithm to determine the optimal caching decisions of RSUs.

- We verify the proposed method with simulations on the Simulation of Urban MObility (SUMO) platform. Simulation results show that the proposed method can better ensure data freshness and low latency.

The remainder of the paper is structured as follows. We introduce the system model in Sect. 2. In Sect. 3, the problem formulation is presented. Our proposed method is detailed in Sect. 4. Section 5 shows the simulation results. Finally, we conclude this paper in Sect. 6.

2 System Model

2.1 HD Map Model

The HD map contains two layers, i.e., the basic layer and the advanced layer. Each layer contains both static and dynamic information, which are used in different driving control functions [9]. Specifically, the basic layer can be used for low-level path planning, and the combination of the basic layer and advanced layer can be used for high-level driving decision-making. Generally, the HD map can be divided into four sub-maps with different data types and update frequency [15], i.e., basic layer with static information f_{bs} and dynamic information f_{bd} , advanced layer with static information f_{as} and dynamic information f_{ad} , as shown in Table 1. Each sub-map is seen as a file, and the file set can be denoted as $\mathcal{F} = \{f_{m,bs}, f_{m,bd}, f_{m,as}, f_{m,ad}\}_{m=1}^M$. Each sub-map f has the same data size denoted as s_f .

Table 1. The components of HD map.

Layer	Data style	Data content	Update frequency
Basic layer	Static landmark data	road facilities, surrounding buildings, trees, etc.	Months
	Dynamic traffic data	Congestion, temporary speed limit, etc.	Seconds or minutes
Advanced layer	Static road data	Road and lane details (curvature, slope, etc.)	Days or months
	Real time environment data	Speed, position and direction of pedestrians and vehicles	Seconds

2.2 System Overview

The IoV network considered in this paper is shown in Fig. 1, which consists of a remote cloud server, M cached-enabled RSUs, and V vehicles scattered around a geographic region. The sets of RSUs and vehicles are denoted as $\mathcal{M} = \{1, 2, \dots, M\}$ and $\mathcal{V} = \{1, 2, \dots, V\}$, respectively. We consider a finite time horizon \mathcal{T} that contains T time slots. The whole region is divided into M blocks, which is covered by one RSU. The set of vehicles that can be served by RSU m at time slot t is denoted as \mathcal{V}_m^t .

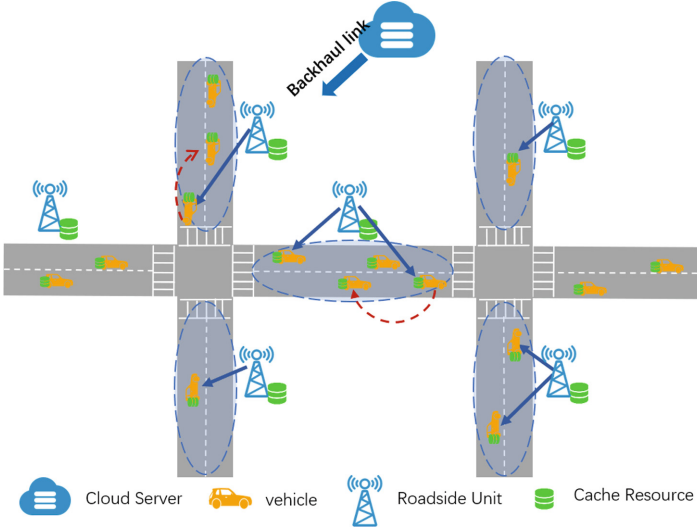


Fig. 1. Vehicle network model.

Each block has an individual HD map, and all the HD maps are stored in the cloud server. The maximum storage space of the RSUs is S . Moreover, RSU m can only communicate with the vehicles located in its own block. In each time slot, RSUs update their caching decision and retrieve the missing sub-maps from the cloud via a high-speed backhaul link. The vehicle fetches the required sub-maps from the cloud server or a RSU via the V2I link, or from the neighboring vehicles via the V2V link. Different spectrum resources are allocated to V2I links and V2V links to avoid interference. Let $\beta^t \in \{0, 1\}^{M \times F}$ denote the caching decision of RSUs in time slot t , where $\beta_{m,f}^t = 1$ means that sub-map f is cached in RSU m in time slot t and $\beta_{m,f}^t = 0$ otherwise. Let $\alpha^t \in \{0, 1\}^{V \times F}$ denote the caching decision of vehicles, where $\alpha_{v,f}^t = 1$ means sub-map f is cached in vehicle v in time slot t and $\alpha_{v,f}^t = 0$ otherwise. The set of vehicles that can communicate with vehicle v in time slot t is denoted as $\mathcal{I}_v^t = \{1, 2, \dots, K\}$.

When vehicle v drives on block m in time slot t , it has to make path planning and driving decisions. The path planning is a low-level control, thus, vehicle v needs to obtain the basic layer f_{bs} and f_{bd} for blocks among all possible paths from its current driving block to its destination block, which is denoted as $\mathcal{F}_{v,l}^t$. Driving decision making is a high-level control, and hence vehicle v needs to obtain basic and advanced layers for its current driving block and possible next arrival block, which is denoted as $\mathcal{F}_{v,h}^t$. In time slot t , the set of sub-maps vehicle v required is $\mathcal{N}_v^t = \mathcal{F}_{v,l}^t + \mathcal{F}_{v,h}^t$. Let \mathcal{U}_v^t denote the set of sub-maps that the vehicle v has already cached locally in time slot t . Thus, the set of sub-maps vehicle v needs to request in time slot t is

$$\mathcal{Q}_v^t = \mathcal{N}_v^t - \mathcal{U}_v^t. \tag{1}$$

We use the age of information (AoI) to characterize the freshness of sub-maps [16]. The AoI of sub-map f in vehicle v in time slot t is denoted as $a_{v,f}^t$, which is defined as the number of time slots since the vehicle v receives dynamic sub-map f until time slot t . Note that for static sub-maps, we have $a_{v,f}^t = 0$. We set a threshold τ for $a_{v,f}^t$. If $a_{v,f}^t$ exceeds τ , sub-map f is considered to be stale and needs to be removed from the vehicle's local cache. Therefore, \mathcal{U}_v^t is updated as

$$\mathcal{U}_v^t = \mathcal{U}_v^{t-1} \bigcup \mathcal{Q}_v^{t-1} - \varepsilon_v^t - (\mathcal{N}_v^{t-1} - \mathcal{N}_v^t), \quad (2)$$

where ε_v^t is the set of stale sub-maps, and $(\mathcal{N}_v^{t-1} - \mathcal{N}_v^t)$ is the set of sub-maps that are no longer needed by vehicle v in time slot t .

2.3 Cost Function

To ensure driving safety, both download latency and freshness loss need to be considered.

1) Download Latency: The V2I bandwidth of RSU m is equally distributed to the vehicles in \mathcal{V}_m^t . Therefore, the download latency of vehicle v fetching a sub-map from RSU m is

$$d_{v,m,f}^t = \frac{sf}{B_{m,v} \log_2 \left(1 + \frac{P_m \cdot g_m}{N} \right)}, \quad (3)$$

where $B_{m,v}$ is the wireless bandwidth between RSU m and vehicle v , P_m is the transmission power of RSU m and g_m is the channel gain. The download latency of vehicle v fetching a sub-map from vehicle k , $k \in \mathcal{I}_v^t$ is

$$d_{v,k,f}^t = \frac{sf}{B_v \log_2 \left(1 + \frac{P_k \cdot g_k}{N} \right)}, \quad (4)$$

where B_v is the bandwidth of the V2V link, P_k is the transmission power and g_k is the channel gain. The download latency vehicle v fetching a sub-map from the cloud server is

$$d_{v,0,f}^t = \frac{sf}{B_0 \log_2 \left(1 + \frac{P_0 \cdot g_0}{N} \right)}, \quad (5)$$

where B_0 is the bandwidth of between the cloud server and vehicle v , P_0 is the transmission power and g_0 is the channel gain.

2) Freshness loss: We define the freshness loss of vehicle v fetching a sub-map f from vehicle k in time slot t as

$$l_{v,k,f}^t = \begin{cases} \frac{a_{k,f}^t}{10\tau}, & \text{if } k \in \mathcal{I}_v^t \text{ and } a_{k,f}^t \leq \tau, \\ 0, & \text{otherwise,} \end{cases} \quad (6)$$

Since sub-maps cached in the cloud server and RSUs are updated in each time slot, thus, the freshness loss via V2I communication is $l_{v,0,f}^t = 0$ and $l_{v,m,f}^t = 0$, $m \in \mathcal{M}$.

In order to strike the balance between these download latency and freshness loss, we define the cost function of vehicle v fetching a sub-map f from the place i as

$$C_{v,i,f}^t = \omega \cdot d_{v,i,f}^t + (1 - \omega) \cdot l_{v,i,f}^t, \quad i \in \{0\} \cup \mathcal{M} \cup \mathcal{V}, \quad (7)$$

where $\omega \in [0, 1]$ is a weight factor. A higher ω value means we emphasize more on the download latency.

3 Problem Formulation

Let $\mathcal{X}^t \in \{0, 1\}^{V \times (M+V+1) \times F}$ denote the request decisions of vehicles in time slot t . $x_{v,k,f}^t = 1$ indicates vehicle v requests sub-map f from vehicle k via V2V communication in time slot t , and $x_{v,k,f}^t = 0$ otherwise. $x_{v,m,f}^t = 1$ indicates vehicle v requests sub-map f from either the cloud server or RSU m via V2I communication in time slot t , and $x_{v,m,f}^t = 0$ otherwise. Therefore, the cost of vehicle v in block m fetches sub-map f in time slot t is

$$D_{v,m,f}^t = x_{v,m,f}^t [\beta_{m,f}^t C_{v,m,f}^t + (1 - \beta_{m,f}^t) C_{v,0,f}^t] + \sum_{k \in \mathcal{I}_v^t} x_{v,k,f}^t \alpha_{k,f}^t C_{v,k,f}^t. \quad (8)$$

Our goal is to select proper caching decision β^t and vehicle request decision \mathcal{X}^t to minimize the total cost, which can be formulated as an optimization problem as

$$\mathbf{P1:} \min_{x^t, \beta^t} \sum_{m=1}^M \sum_{v \in \mathcal{V}_m^t} \sum_{f \in \mathcal{Q}_v^t} D_{v,m,f}^t$$

$$\text{s.t.} \quad x_{v,k,f}^t \leq \alpha_{k,f}^t, \quad (9a)$$

$$\sum_{k \in \mathcal{I}_v^t} x_{v,k,f}^t + x_{v,m,f}^t = 1, \quad (9b)$$

$$\sum_{f \in \mathcal{F}} \beta_{m,f}^t s_f \leq S, \quad (9c)$$

$$\beta_{m,f}^t \in \{0, 1\}, x_{v,k,f}^t, x_{v,m,f}^t \in \{0, 1\}, \quad (9d)$$

Constraint (9a) indicates that vehicle v cannot request sub-map f from vehicle k without sub-map f . Constraint (9b) means that vehicle v can only get one copy of sub-map f . Constraint (9c) indicates the size of all the cached maps in RSU m should not exceed its maximum storage capacity.

4 Freshness-Aware HD Map Caching Algorithm

It is intractable to solve **P1** directly due to its huge solution space and tight coupling between the caching decision and the vehicle request decision. Therefore, in the following, we decompose the original problem into two subproblems:

the vehicle request problem and the caching placement problem. First, given a fixed caching strategy β^t , we derive an efficient algorithm to obtain the optimal \mathbf{x}_o^t . Next, we solve the caching placement problem with distributed MAMAB algorithm based on the obtained request decision \mathbf{x}_o^t .

4.1 Freshness-Aware Vehicle Request Algorithm

We assume that RSUs cache all the HD sub-maps and make decisions in each time slot, i.e., $\beta^t = \mathbf{1}$, then the cost of vehicle v fetches f in time slot t can be simplified as

$$D'_{v,m,f} = x_{v,m,f}^t C_{v,m,f}^t + \sum_{k \in \mathcal{I}_v^t} x_{v,k,f}^t \alpha_{k,f}^t C_{v,k,f}^t. \quad (9)$$

Therefore, the optimization problem can be rewritten as follows:

$$\begin{aligned} \mathbf{P2}: \min_{\mathbf{x}^t} & \sum_{m=1}^M \sum_{v \in \mathcal{V}_m^t} \sum_{f \in \mathcal{Q}_v^t} D'_{v,m,f} \\ \text{s.t.} & \quad (9a), (9b), \\ & \quad x_{v,k,f}^t, x_{v,m,f}^t \in \{0, 1\}. \end{aligned} \quad (10)$$

Note that **P2** is an integer linear programming (ILP) problem, which can be solved by conventional ILP algorithms, e.g., branch and bound (B&B) algorithm. However, there are $k + 1$ possible request cases for each sub-map f , thus, the worst-case time complexity of B&B algorithm is $\mathcal{O}\left((k + 1)^{|\mathcal{Q}_v^t|}\right)$.

In order to reduce complexity, we solve **P2** with a matching-based freshness-aware vehicle request algorithm, which is detailed in Algorithm 1. The proposed algorithm works as follows. First, for vehicle v in the coverage area of RSU m , we calculate the cost of fetching sub-map f from the RSU m , i.e., $C_{v,m,f}^t$, and that from vehicle k if $\alpha_{k,f}^t = 1$, i.e., $C_{v,k,f}^t$. Then, for vehicle k , we obtain the difference of the costs $\Delta L_k^t = C_{v,m,f}^t - C_{v,k,f}^t$, $k \in \mathcal{I}_v^t$. If $\Delta L_k^t > 0$, vehicle k will be added to the candidate set ψ_v^t of vehicle v . The vehicles in ψ_v^t are ranked in descending order by the cost difference ΔL_k . If $\psi_v^t = \emptyset$ or $\Delta L_k \leq 0$, we set $x_{v,m,f}^t = 1$, which means vehicle v fetches sub-map f from RSU. Finally, we find the maximum cost difference in the candidate set ψ_v^t and determine whether vehicle v fetches sub-map from the corresponding vehicle according to the average cost of vehicles on the block m , until convergence.

4.2 Distributed MAMAB-Based Caching Algorithm

Compared with fetching a sub-map from the cloud, the delay of fetching a sub-map from the RSU is smaller. For a sub-map f , the more vehicles in block m fetch it via V2I, the more cost will be reduced by caching it at RSU m . We define the reward of RSU m caching sub-map f at time slot t as

$$r_{m,f}^t = \sum_{v \in \mathcal{V}_m^t} \mathbb{I}(f, \mathcal{G}_{v,m}^t), \quad (11)$$

Algorithm 1. Freshness-Aware Vehicle Request Algorithm**Input:** B_0, B_v, B_m and vehicle caching variable α^0 **Output:** vehicle request decision

- 1: Initialize the vehicle request variable.
- 2: **for** $m = 1, 2, \dots, M$ **do**
- 3: Calculate the cost for vehicle $v \in V_m^t$ to download sub-map from RSU m and each vehicle in I_v^t .
- 4: **while** the search for V_m^t or I_v^t is not complete **do**
- 5: Calculate the difference of the costs $\Delta L_k = C_{v,m,f}^t - C_{v,k,f}^t$.
- 6: Vehicles that satisfy (9a) and $\Delta L_k > 0$ will be included in ψ_v .
- 7: Rank the elements of ψ_v in descending order by ΔL_k .
- 8: Find the maximum cost difference in ψ_v of vehicle v .
- 9: Update total cost $C_{total}(w)$ of vehicles in V_m^t fetching sub-map in iteration w .
- 10: **if** $C_{total}(w) \leq C_{total}(w - 1)$ **then**
- 11: Update the preference list of vehicle v
- 12: **else**
- 13: Save the vehicle request decision for iteration $w - 1$.
- 14: **end if**
- 15: **end while**
- 16: **end for**
- 17: **return** the vehicle request decision

where $\mathcal{G}_{v,m}^t$ means the sub-maps that vehicle v needs to request from RSU m in time slot t . $\mathbb{I}(f, \mathcal{G}_{v,m}^t) = 1$ if $f \in \mathcal{G}_{v,m}^t$, and $\mathbb{I}(f, \mathcal{G}_{v,m}^t) = 0$ otherwise. Instead of minimizing the total cost, we reformulate the caching problem into a MAMAB problem with the aim of maximizing the total reward, which is defined as

$$\begin{aligned}
 \mathbf{P3}: \max_{\beta^t} & \sum_{t=1}^T \sum_{m=1}^M \sum_{f \in \mathcal{F}} \beta_{m,f}^t r_{m,f}^t & (12) \\
 \text{s.t.} & (9c), \\
 & \beta_{m,f}^t \in \{0, 1\}.
 \end{aligned}$$

As the number of RSUs increases, the action space will be too large, resulting in additional delay by the traditional centralized approach. Additionally, there is extra communication overhead between RSUs and the central controller. Thus, we adopt a distributed MAMAB approach in which each RSU is considered as an agent and the sub-maps are arms. The Upper Confidence Bound (UCB) method is adopted to obtain the tradeoff between exploration and exploitation. First, each RSU m caches sub-maps randomly until each sub-map f is cached once. Then we update the number of times that sub-map f is cached in RSU m , which is denoted as $J_{m,f}^t$. Define the average reward of caching sub-map f in RSU m

Algorithm 2. Distributed MAMAB-Based Caching Algorithm**Input:** vehicle request decision x^t , \mathcal{G}_v^t **Output:** cache decisions of RSUs

- 1: Each RSU caches sub-maps randomly until each sub-map is cached once.
- 2: Update the number of times sub-map f cached in RSU m $J_{m,f}^0 = 1$.
- 3: **for** $t = 1, 2, \dots, T$ **do**
- 4: **for** $m = 1, 2, \dots, M$ **do**
- 5: Calculate estimated reward $\hat{R}_{m,f}^t$ of caching sub-map f .
- 6: Calculate the real reward $r_{m,f}^t$ of caching sub-map f at RSU m .
- 7: RSU m optimizes caching strategy by $\max \sum_{f=1}^F \beta_{m,f}^t \hat{R}_{m,f}^t$ with constraint (9c).
- 8: Update the average caching reward according to the cache strategy.
- 9: **if** $\beta_{m,f}^t = 1$ **then**
- 10: $\bar{R}_{m,f}^t = \frac{\bar{R}_{m,f}^{t-1} J_{m,f}^{t-1} + r_{m,f}^t}{J_{m,f}^{t-1} + 1}$ and $J_{m,f}^t = J_{m,f}^{t-1} + 1$.
- 11: **else**
- 12: $\bar{R}_{m,f}^t = \bar{R}_{m,f}^{t-1}$ and $J_{m,f}^t = J_{m,f}^{t-1}$.
- 13: **end if**
- 14: **end for**
- 15: **end for**
- 16: **return** the cache decisions of RSUs

at time slot t as $\bar{R}_{m,f}^t$. We define the estimated reward as

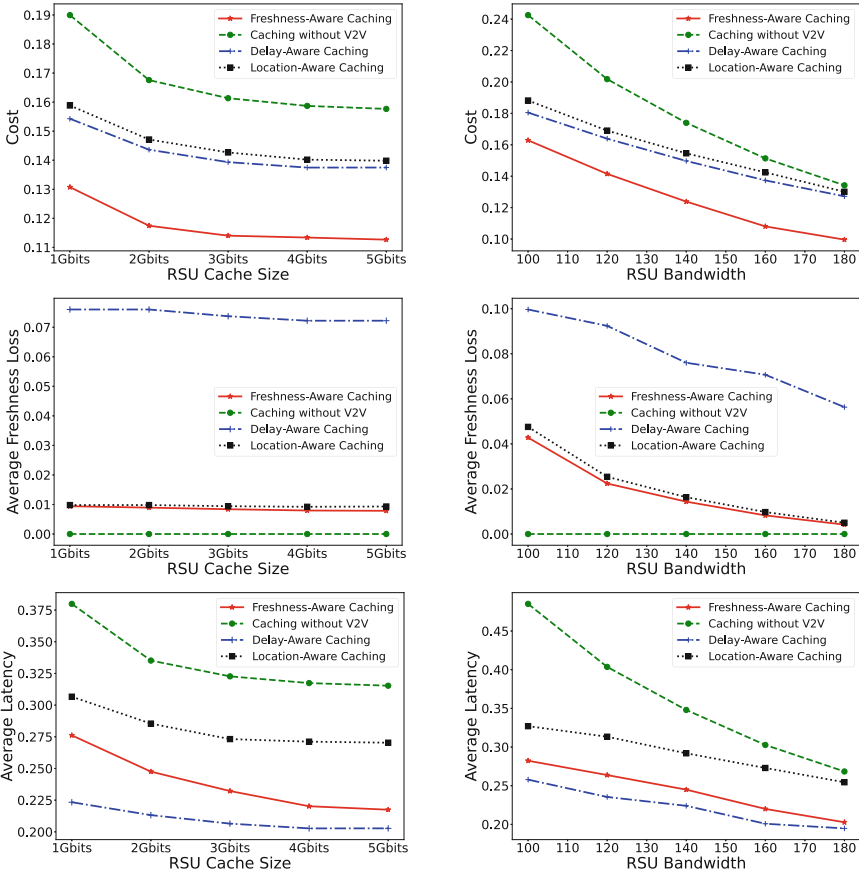
$$\hat{R}_{m,f}^t = \bar{R}_{m,f}^{t-1} + \sqrt{\frac{3 \log(\gamma_f^2 t)}{2J_{m,f}^{t-1}}}, \quad (13)$$

where γ_f^2 is the upper bound on the reward of RSU m caching sub-map f . When the number of explorations of the sub-map f is small, $J_{m,f}^t$ will be small.

Thus, $\sqrt{\frac{3 \log(\gamma_f^2 t)}{2J_{m,f}^{t-1}}}$ is large, and the probability of sub-map f being selected will be high. This is the exploration process of sub-map f . When each sub-map has been explored enough, the value of $J_{m,f}^t$ is large, and the main influence is concentrated on $\bar{R}_{m,f}^t$, which is the exploitation of sub-map f . Each RSU m optimizes its caching strategy by $\max \sum_{f=1}^F \beta_{m,f}^t \hat{R}_{m,f}^t$ with the cache size constraint. And then, update the average reward according to the cache strategy. If $\beta_{m,f}^t = 1$, $\bar{R}_{m,f}^t = \frac{\bar{R}_{m,f}^{t-1} J_{m,f}^{t-1} + r_{m,f}^t}{J_{m,f}^{t-1} + 1}$ and $J_{m,f}^t = J_{m,f}^{t-1} + 1$. Otherwise, $\bar{R}_{m,f}^t = \bar{R}_{m,f}^{t-1}$ and $J_{m,f}^t = J_{m,f}^{t-1}$. The details of the proposed algorithm are shown in Algorithm 2.

5 Simulation Results

In this section, we evaluate the performance of the proposed method with Simulation of Urban MObility (SUMO) software. We consider a 1600×1600 m²



(a) The impact of RSU cache size. (b) The impact of RSU bandwidth.

Fig. 2. The total cost, average freshness loss and average download latency of different methods with under the impact of (a) the RSU cache size and (b) the RSU bandwidth, respectively.

grid road network with $K=40$ RSUs. Therefore, there are 160 sub-maps. The size of each sub-map is $s_f = 100$ Mbits and the cache size of each RSU $S = 2$ Gbits [9]. The bandwidth of a vehicle and the RSU are set to 50 and 150 MHz, respectively. The communication range of a vehicle is set to 100 m. In addition, the transmission power of a vehicle and RSUs are 300 mW and 2 W, respectively. The gaussian channel noise and channel gain are set to 10^{-6} mW and 5 dB, respectively [17]. There are $T = 20000$ slots and the threshold $\tau = 5$ time slots. We set $\omega = 0.5$. The following benchmarks are used for comparison.

- **Caching without V2V** [13]: each RSU caches sub-maps to maximize the average caching reward without V2V collaboration. Distributed MAMAB method is utilized to make the caching decisions.

- **Delay-Aware Caching** [11]: vehicle request decisions are made to minimize the download latency without considering the freshness of dynamic sub-maps. Both V2I and V2V communications are considered.

- **Location-Aware Caching**: each RSU caches sub-maps of the blocks from the near to far until its cache size is full. The vehicle request decisions are determined by Algorithm 1.

Figure 2(a) shows the total cost, average freshness loss and average download latency of all the methods with the RSU cache size varies from 1 Gbits to 5 Gbits. As can be seen from the figure, the average freshness loss of caching without V2V is the lowest, while its download latency is the highest. This is because all the sub-maps are fetched from RSUs or the cloud platform. On the contrary, the average freshness loss of delay-aware caching is the highest, while its download latency is the lowest. This is because this method only considers download latency when making vehicle request decisions. The average freshness loss of location-aware caching is close to that of caching without V2V, which proves the effectiveness of Algorithm 1. However, its average download latency is the second highest, due to the fact that its caching decision is not optimized. The proposed freshness-aware caching method achieves a trade-off between the average freshness loss and the average download latency, and hence its total cost is the lowest. We also observe that as the RSU cache size increases, the total cost of all the methods decreases since more sub-maps can be fetched from the RSUs.

The total cost, average freshness loss, and average download latency of the four methods with different RSU bandwidth is shown in Fig. 2(b). Similar results can be obtained which is consistent with that in Fig. 2(a). Again, the total cost of the proposed freshness-aware caching method due to the same reason presented above. Moreover, as the RSU bandwidth increases, the download latency of obtaining sub-maps via V2I is reduced, which can further reduce the total cost.

6 Conclusion

In this paper, we proposed a freshness-aware HD map caching method to minimize both download latency and freshness loss in IoV. We introduced a cost function to incorporate both factors and then formulated the caching problem as an optimization problem to minimize the total cost. First, we designed a freshness-aware vehicle request algorithm to optimize vehicle request decisions. Next, distributed MAMAB algorithm is leveraged to make optimal caching decisions for RSUs. Simulation results demonstrated that the proposed method outperforms other caching methods. In practical scenarios, not all the vehicles are willing to share their cache resources, therefore, a proper incentive mechanism is needed to encourage vehicles to participate in the HD map caching and sharing process. This is left as a future study.

Acknowledgements. This work is support in part by National Key R&D Program of China under Grant 2019YFB2102400, in part by NSFC Joint Funds under Grant

U2001204, in part by the National Natural Science Foundation of China under Grant 62072330, and also in part by the Natural Science Foundation of Tianjin under Grant 20211093.

References

1. Seif, H.G., Hu, X.: Autonomous driving in the iCity-HD maps as a key challenge of the automotive industry. *Engineering* **2**(2), 159–162 (2016)
2. Adachi, JM.: Accuracy of global navigation satellite system based positioning using high definition map based localization. U.S. Patent 10,527,734 (2020)
3. Paschos, G.S., Iosifidis, G., Tao, M., et al.: The role of caching in future communication systems and networks. *IEEE J. Sel. Areas Commun.* **36**(6), 1111–1125 (2018)
4. Breslau, L., Cao, P., Fan, L., Phillips, G., Shenker, S.: Web caching and Zipf-like distributions: evidence and implications. In: *IEEE Conference on Computer Communications (INFOCOM)*, pp. 126–134. IEEE, New York (1999)
5. Su, Z., Hui, Y., Xu, Q., Yang, T., Liu, J., Jia, Y.: An edge caching scheme to distribute content in vehicular networks. *IEEE Trans. Veh. Technol.* **67**(6), 5346–5356 (2018)
6. Wang, R., Kan, Z., Cui, Y., Wu, D., Zhen, Y.: Cooperative caching strategy with content request prediction in internet of vehicles. *IEEE Internet Things J.* **8**(11), 8964–8975 (2021)
7. Yu, Z., Hu, J., Min, G., Zhao, Z., Wang, M., Hossain, M.S.: Mobility-aware proactive edge caching for connected vehicles using federated learning. *IEEE Trans. Intell. Transp. Syst.* **22**(8), 5341–5351 (2020)
8. Zhao, Z., Guardalben, L., Karimzadeh, M., et al.: Mobility prediction-assisted over-the-top edge prefetching for hierarchical VANETs. *IEEE J. Sel. Areas Commun.* **36**(8), 1786–1801 (2018)
9. Xu, X., Gao, S., Tao, M.: Distributed online caching for high-definition maps in autonomous driving systems. *IEEE Wireless Commun. Lett.* **10**(7), 1390–1394 (2021)
10. Yang, L., Zhang, L., He, Z., Cao, J., Wu, W.: Efficient hybrid data dissemination for edge-assisted automated driving. *IEEE Internet Things J.* **7**(1), 148–159 (2020)
11. Huang, X., Xu, K., Chen, Q., Zhang, J.: Delay-aware caching in internet-of-vehicles networks. *IEEE Internet Things J.* **8**(13), 10911–10921 (2021)
12. Liu, J., Zhang, C., Wang, Y., Wei, L., Liu, J.: Cooperative caching in a content-centric network for high-definition map delivery. In: *2020 3rd International Conference on Hot Information-Centric Networking (HotICN)*, pp. 96–101, Hefei (2020)
13. Wu, Y., Shi, Y., Li, Z., Chen, S.: A cluster-based data offloading strategy for high definition map application. In: *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, pp. 1–5. IEEE, Antwerp (2020)
14. Wang, F., Guan, D., Zhao, L., Zheng, K.: Cooperative V2X for high definition map transmission based on vehicle mobility. In: *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, pp. 1–5. IEEE, Kuala Lumpur (2019)
15. Liu, R., Wang, J., Zhang, B.: High definition map for automated driving: overview and analysis. *J. Navig.* **73**(2), 324–341 (2020)
16. Kaul, S., Yates, R., Gruteser, M.: Real-time status: how often should one update? In: *2012 Proceedings IEEE INFOCOM*, pp. 2731–2735. IEEE, Florida (2012)
17. Dai, P., Hu, K., Wu, X., et al.: Asynchronous deep reinforcement learning for data-driven task offloading in MEC-empowered vehicular networks. In: *IEEE Conference on Computer Communications (INFOCOM)*, pp. 1–10. IEEE, Vancouver (2021)



A Scalable Blockchain-Based Trust Management Strategy for Vehicular Networks

Minghao Li, Gansen Zhao^(✉), and Ruilin Lai

School of Computer Science, South China Normal University,
Guangzhou 510631, China
gzhao@m.scnu.edu.cn

Abstract. In recent years, the dynamic environment on the Internet of Vehicles led vehicular communication networks' trust management mechanism to become a research hotspot. Most of the existing trust management in vehicular networks relies on a centralized third party. However, it causes trust management to be limited to a single node and has high requirements for device performance. In order to improve the reliability of information exchanged between vehicles, we propose a scalable blockchain-based trust management strategy, which employs vehicle-related objective factors to evaluate the credibility of information transmitted between vehicles to determine the vehicles' trust level. We also design a consensus mechanism to make all RSUs (Roadside Units) maintain a consistent and reliable distributed ledger as nodes so that vehicles can obtain global trust information more quickly during interaction to improve its reliability. The security and performance analysis shows that our strategy has high reliability and availability.

Keywords: Blockchain · Vehicular networks · Trust management · Consensus mechanism

1 Introduction

With the development of 5G technology and smart car, the Internet of Vehicles (IoV) has undergone a huge transformation. In today's vehicular networks, vehicles obtain various information such as road or surrounding vehicle conditions by communicating with surrounding vehicles (V2V, Vehicles to Vehicles) or communicating with surrounding infrastructures (V2I, Vehicles to Infrastructures) to improve the efficiency and safety of vehicles driving [1]. Ensuring secure and trusted communication in vehicular networks is complex due to numerous uncertain threats, such as low-quality and false information.

Due to the high variability of the vehicular networks' topology, the vehicles that conduct V2V communication are likely to be completely unfamiliar. The reliability of messages sent between strange vehicles cannot be ensured, which may lead to the spread of false information and cannot be resolved. It will pose a

threat to the traffic situation [2]. Therefore, it is necessary to propose an effective trust model to improve the reliability of the interaction between vehicles.

The current trust models applied to vehicular networks are divided into entity-oriented (vehicle), data-oriented (message), and hybrid trust models according to different evaluation objects [3]. Even a high-confidence entity can threaten the system if the message sent is fake because the entity may send fake messages for selfish or malicious purposes [4]. Hence, it is necessary to evaluate the message's authenticity. In many state-of-the-art solutions, the vehicle judges whether the information received is trustworthy, providing a basis for network operators to reward and punish specific vehicles [5].

Existing trust management strategies are roughly divided into centralized and decentralized. For the centralized [6], trust management tends to rely too much on a single authority or infrastructure, and this causes many problems in terms of communication costs and data tampering or leakage. For the decentralized [7], the RSU acts as the role of computing and synchronizing data. However, this also leads to new problems, such as the inconsistent trust data affecting the quality of V2V service.

Blockchain provides a decentralized distributed ledger that allows multiple parties that do not trust each other to conduct transactions without the need for a central third party, ensuring the transparency and traceability of transaction data while protecting the integrity and security of data. In vehicular networks, RSUs are usually distributed, and the blockchain is maintained through multiple RSUs to synchronize information to achieve trust management. Any single transaction is verified by entities in the P2P (Peer-to-Peer) network, then packaged into blocks and updated to the chain according to the established consensus mechanism [8]. When there are malicious vehicles or faulty RSUs, trust evaluation and consensus mechanism ensure the system's normal operation. Consensus mechanisms may also face problems affecting the system, such as blockchain forks, high latency caused by excessive computing power requirements, and monopolizing miners' election results. Therefore, a reliable consensus mechanism that conforms to the vehicular network and enables multiple nodes to maintain trust data consistently is required, combined with a trust evaluation method that improves trust management in reliability and availability. The main contributions are summarized as follows:

- A decentralized and scalable trust management framework in vehicular networks, vehicles make use of objective factors (time, location, historical trust) to evaluate the credibility of each other. The evaluation results are uploaded to the RSU to calculate and update the vehicles' trust value.
- A joint Proof-of-Stake (PoS) and Practical Byzantine Fault Tolerance (PBFT) consensus mechanism, all RSUs in a certain area jointly maintain a consortium blockchain, some RSUs will validate the block, and all RSUs synchronize the trust data according to the consensus mechanism.
- Evaluating the security properties and performance of the trust management strategy proves that the scheme has high reliability and availability.

2 Related Work

Utilizing blockchain technology with a consensus mechanism can solve the problems with centralize and decentralize. Articles combining trust management with blockchain usually rely on multiple RSUs or edge servers to maintain the trust information. In [5], RSUs calculate the vehicles’ trust value offset and follow the joint PoS and PoW consensus mechanism to update the trust information. That is to say, the greater the variation of the vehicles’ trust value offset in the area, the easier it is for RSU to complete the PoW (find the nonce) and update the block. In [9], the certificate issuance and revocation mechanism are used for vehicles. RSUs follow a dynamic PoW consensus mechanism. The higher the traffic flow in the coverage area, the easier it is to find the nonce to reach a consensus. The authors of [10] proposed that RSUs globally maintain the trust scores calculated by adjacent vehicle nodes with each other through the PBFT protocol.

However, the above articles cause new problems. Some transactions are not guaranteed to update the blockchain successfully due to forks. The hash value computing also has high requirements on the device. The current consensus mechanism may have a high delay due to the consensus reached after multi-party verification. Besides, the equipment with high performance may monopolize the right to produce blocks in the blockchain network. The above problems may lead to data loss or tampering.

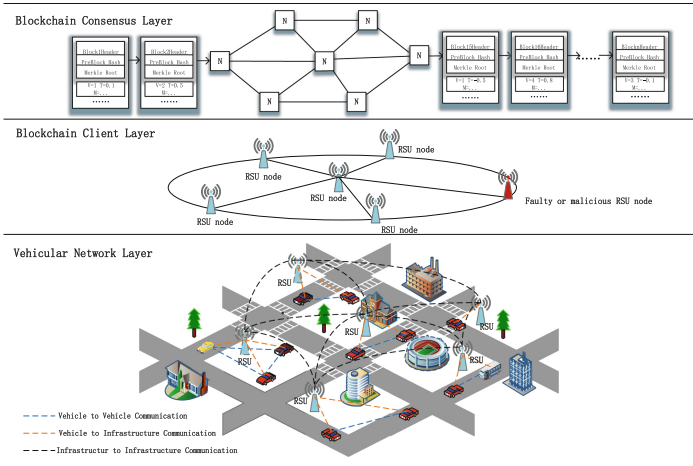


Fig. 1. Overall framework

3 System Model

3.1 Overall Framework

The overall framework of our proposed vehicular network trust management is shown in Fig. 1. The functions of each part are as follows:

Vehicle. Participants in the vehicular networks communicate with surrounding vehicles and RSUs. The vehicle senses the surrounding traffic conditions or events and sends them to surrounding vehicles through V2V communication.

RSU. It is mainly responsible for 1) collecting the evaluation results of messages of surrounding vehicles from the receiving vehicle and 2) calculating the surrounding vehicles' trust value according to the evaluation results. 3) Several RSUs in a certain range maintain the blockchain to ensure data consistency.

Besides, the overall framework of this paper consists of three layers:

Vehicular Network Layer. This layer consists of RSUs and vehicles equipped with onboard sensors, computing and communication equipment. When a vehicle enters the communication range of an RSU, it interacts with the RSU as a user node. Trust information is stored in RSUs. The vehicle obtains the surrounding vehicles' trust information by accessing the RSU and employs the trust information to determine whether the vehicle is a trusted entity.

Blockchain Client Layer. RSUs are both clients and consensus nodes that participate in the consensus. The client RSU requests consensus after packaging the multiple vehicles' trust value in the coverage area for a period of time.

Blockchain Consensus Layer. According to our proposed joint PoS and PBFT consensus mechanism, the trust information block is validated by validator nodes selected according to the stake and then updated to the blockchain.

3.2 Overall Flow

Figure 2 shows the overall flow of trust management for vehicular networks. First, vehicle A requests to communicate with vehicle B, and vehicle B queries the historical trust value of vehicle A from RSU and the blockchain. After RSU returns the trust value, vehicle B judges whether it can communicate with vehicle A according to the trust value. If possible, vehicle A sends a message to vehicle B. Then, vehicle B evaluates the message's credibility and uploads the evaluation result to RSU. Next, according to the evaluation results, RSU calculates vehicle A's trust value. Finally, an RSU packages multiple vehicles' trust value to initiate a consensus request.

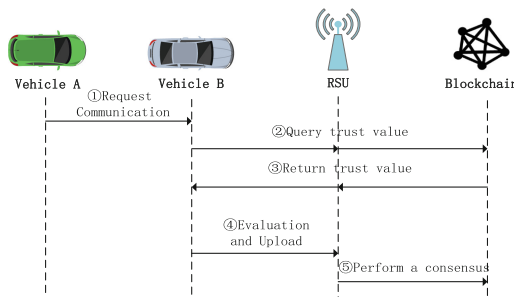


Fig. 2. The overall flow between vehicles, Rsu and blockchain in an information service

4 Detailed Methodology

4.1 Message Credibility Evaluation

After the vehicle receives messages from the surrounding vehicles, the messages' credibility is evaluated. First, the message sent by the sender will contain information such as $ID_i, ID_j, Report_n, addr_i, time_i$. ID_i, ID_j are the unique identifiers of message sender and receiver, respectively. $Report_n$ is a related report on event n . $addr_i$ is the location where the vehicle sends the message, and $time_i$ is the timestamp when it sent the message. Each message receiver maintains a set E of event reports, in which m events described in different messages are divided into $\langle E_1, E_2, \dots, Em \rangle$ according to the messages sent by surrounding vehicles, each subset E_n contains reports $\langle r_n^1, r_n^2, \dots, r_n^a \rangle$ of events n from surrounding a vehicles. As the sender who reports the event closer to the place and time of the event happens has higher credibility, the messages sent by vehicles with higher historical trust value are more trustworthy. Therefore, we consider three factors (distance, time, historical trust value) to calculate the message's credibility of surrounding vehicles in Eq. (1).

$$\begin{aligned}
 e_n^b(\Delta d, \Delta t, T) &= e^{Distance+Time+curTrust} \\
 \text{s.t. } Distance &= \mu_1 - \omega_1 |\sin(\Delta d)| \\
 Time &= \omega_2 |\Delta t| \\
 curTrust &= \omega_3 T
 \end{aligned} \tag{1}$$

where $e_n^b \in (0, 1]$, denotes the credibility of the message reported by vehicle b to event n . Δd is the distance between the message sender and the incident location, Δt indicates the time difference between the vehicle reported and the event occurred, T represents the current trust value of message sender, and $\omega_1, \omega_2, \omega_3$ represent the their respective weights. Through the calculation, the credibility set $e_n = \langle e_n^1, e_n^2, \dots, e_n^a \rangle$ of reports of event n sent by multiple surrounding vehicles is obtained, and the event's credibility α is also subsequently obtained by combining with Bayesian Inference [5, 11] as follows:

$$P[\alpha|e] = \frac{P[\alpha] \cdot \prod_{b=1}^N P[e^b|\alpha]}{P[\alpha] \cdot \prod_{b=1}^N P[e^b|\alpha] + P[\bar{\alpha}] \cdot \prod_{b=1}^N P[e^b|\bar{\alpha}]} \tag{2}$$

where $P[\alpha|e] \in [0, 1]$, $\bar{\alpha}$ is the mutually exclusive event of event α , $P[e^b|\alpha]$ is the probability that sender b confirms event α when α occurs, and $P[\alpha]$ is the prior probability that event α occurs. We assume that an event is true only when $P[\alpha|e]$ is greater than 0.5; otherwise, it is considered fake. According to Eq. (2), the evaluation result of the event's authenticity is obtained. Subsequently, it concluded whether the event reports sent by surrounding vehicles were credible.

4.2 Trust Value Calculation

When a message receiver completes the evaluation, the evaluation information was packaged in the form of $(ID_i, ID_j, Report_n, credible/incrredible)$, where *credible/incrredible* represents the evaluation result of report $Report_n$. Then, upload information to the nearby RSU, which recalculates the vehicle's trust value according to Algorithm 1. If a vehicle's trust value is not less than V_{high} , or it broadcasts unreal messages maliciously many times, it will suffer a heavier punishment and need to spread more real information to make up.

Algorithm 1 Trust value calculation

Input: $(ID_i, ID_j, Report_n, credible/incrredible)$,
 $curT_j$, the current trust value of vehicle j ;

Output: T_j , the calculated trust value of vehicle j ;

1: **if** $Report_n \Rightarrow credible$ **then**

$$T_j = \begin{cases} Exit & curT_j == V_{upperbound} \\ reward(curT_j) & curT_j \in [V_{low}, V_{upperbound}) \end{cases}$$

2: **else if** $Report_n \Rightarrow incrredible$ and less than 3 times **then**

$$T_j = \begin{cases} Exit & curT_j == V_{upperbound} \\ p.high(curT_j) & curT_j \in [V_{high}, V_{upperbound}) \\ p.mid(curT_j) & curT_j \in [V_{low}, V_{high}) \\ p.low(curT_j) & curT_j \in [V_{low}, V_{high}) \end{cases}$$

3: **else if** $Report_n \Rightarrow incrredible$ and not less than 3 times **then**

$$T_j = \begin{cases} Exit & \\ p.high(curT_j) & curT_j \in [V_{low}, V_{upperbound}] \end{cases}$$

4: **else** Exit;

5: **return** T_j ;

4.3 Consensus Mechanism

This section chooses to select validator nodes based on stake. The RSUs with the fewest vehicles in the coverage area are selected as validators to participate in the block validation. In other words, the traffic flow within the scope is used as the stake to select validator nodes. In this way, RSUs with many vehicles in the coverage area have to allocate resources to participate in the block validation while performing calculations and communication can be avoided, improving block generation efficiency. More importantly, reducing the number of validator nodes avoids the low communication efficiency and high cost caused by excessive validator nodes in the PBFT [12]. Similarly, using a highly variable objective factor (traffic flow) as a stake to select validator nodes prevents the election results from being monopolized by the adversary who controls multiple shareholders in the PoS [13]. The main steps of our proposed consensus mechanism are as follows:

Validator Selection. First, the leader node is selected from n nodes in the entire network. The leader sorts and broadcasts all the requests from client RSU and packages trust information into blocks to initiate consensus. Then, according

Algorithm 2 The selection of validators

Input: R_{n-1} , the set of each RSU nodes participating in the consensus;
 V_{n-1} , the set of the number of vehicles covered by each RSU node;
Output: R_m , m RSU nodes with least vehicle in coverage range, $R_m \in R_{n-1}$;
Step1: Calculate the number of validator node
 1: $m = \max(\log(\xi \cdot (n - 1)), m_{min})$;
Step2: Select m validator nodes
 1: **for** V_{n-1}, R_{n-1} **do**
 2: generate V_m , contains the smallest m values in V_{n-1} ;
 3: choose R_m from R_{n-1} based on V_m ;
 4: **return** R_m

to Algorithm 2, the number of validator nodes m to be selected is determined from the remaining $n - 1$ RSU nodes. Next, according to the stake (traffic flow), the m RSUs with the smallest traffic flow are selected as the validator nodes.

Request. A client RSU sends a request message to the leader, which contains the vehicles’ trust information in the coverage of RSU.

Pre-prepare. When the leader receives the request message from the client RSU, it generates a *pre - prepare* message and broadcasts it to the previously selected validator nodes. Then, the validator node receives and validates the *pre - prepare* message from the leader node. If passed, it broadcasts the *prepare* message to other validator nodes and then enters the *prepare* phase.

Prepare. A validator node receives the *prepare* message and validates it. If the node receives $\frac{2}{3}m + 1$ correct messages, it broadcasts the *commit* message and enter the *commit* phase.

Commit. A validator node receives the *commit* message from other validator nodes and validates it. If the node receives over $\frac{2}{3}m + 1$ valid *commit* messages, the block validation completed. Each validator node synchronizes the validated block information to the remaining $n - m - 1$ RSUs that do not participate in the validation, and the block is updated to the blockchain. Then reply the *reply* message to the client. The block validation process is shown in Fig. 3.

Reply. After the client receives $\frac{1}{3}m + 1$ *reply* messages with the same timestamp and response value, it considers the response to be successful.

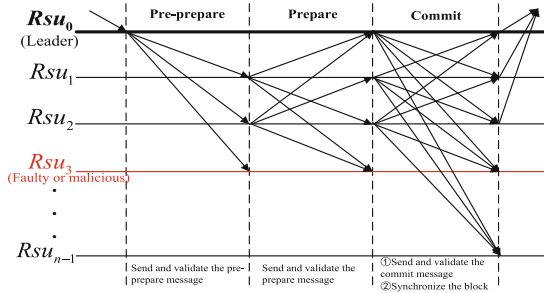


Fig. 3. The validation of block information

5 Security Analysis

This part mainly introduces the security properties based on the possible threat model in the strategy.

Message Sender

- *Message spoofing*: The message sender sends randomly generated, counterfactual, or low-quality messages to the receiver to reduce driving safety. However, in this strategy, the receiver evaluates the events' credibility by calculating the credibility of reports issued by multiple surrounding vehicles.
- *On-off*: The sender may sometimes send real or false messages, making it difficult to identify its threats. However, repeated acts of evil result in heavier punishments in this strategy. The message sender must first send some real messages to make its trust value higher than the set threshold, which increases the cost of doing evil.

Message Receiver

- *Bad mouthing*: The message receiver may maliciously generate false evaluation results and upload them to the RSU. However, the message sender sends the message to each surrounding vehicle. Under the premise that the number of attackers is limited, the wrong evaluation result only has a small influence on the system.

Roadside Unit

- *Faulty RSU*: Suffering from DoS attack, network intrusion, or physical damage caused RSU failure, making it unable to participate in the consensus mechanism. However, according to our proposed consensus mechanism, the system is able to tolerate partial faulty or malicious RSU nodes.

6 Performance Analysis

This section will evaluate the performance of the proposed blockchain trust management framework. We first use Python (version 3.9.4) programming language to compare and analyze the influence of time and distance factors on the message's credibility in the trust evaluation. Then we simulate the change of trust value between honest vehicles and malicious vehicles by implementing simulation. The trust value calculation utilizes smart contracts written in Solidity (a high-level programming language created to implement smart contracts). Finally, the performance of the proposed blockchain consensus mechanism is evaluated through the FISCO BCOS (version 2.8.0) consortium blockchain platform.

6.1 Message Credibility

This part mainly elaborates on the influence of time and distance factors on the message's credibility in the evaluation stage through Eq. 1. Figure 4(a) and 4(b) show the impact of the different distances and time differences on the message's

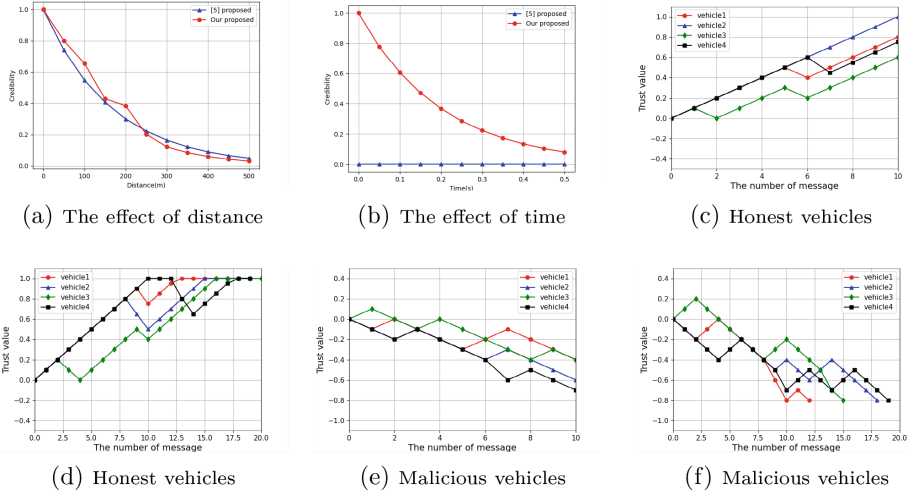


Fig. 4. The influence of objective factors on credibility and the changing trend of vehicles' trust value

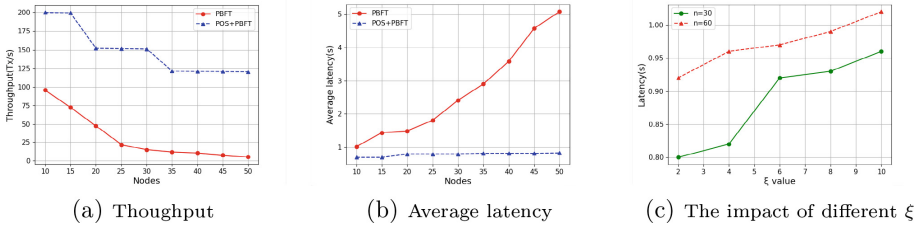


Fig. 5. Blockchain performance test

credibility, respectively. It can be seen from Fig. 4(a) that the farther away from the accident site, the lower the credibility of the message sent by the vehicle. Compared with [5], when the distance is less than 230 m, our method's credibility result is higher. On the contrary, it is lower when the distance exceeds 230 m. Our approach more precisely achieves that the closer the distance, the more credible the reported message. As shown in Fig. 4(b), the longer it's been since the incident happened, the less credible it is. The above results show that our method is more fine-grained, improving the accuracy of message credibility evaluation and trust evaluation.

6.2 Trust Value Calculation

Trust value calculation requires distinguishing honest vehicles and malicious vehicles effectively. As shown in Fig. 4(c), 4(d), 4(e), and 4(f), the changes in trust value of honest vehicles and malicious vehicles after 10 and 20 interactions can be obtained respectively. With the increase in the number of interactions, the

honest vehicles' trust value shows a significantly increased trend in Fig. 4(c), 4(d). In contrast, the malicious vehicles' trust values show a clear downward trend in Fig. 4(e), 4(f). Most of the messages sent by honest vehicles are real but may also send wrong messages, resulting in fluctuations in their trust value. As malicious vehicles may also send real information to avoid being identified, the malicious vehicles' trust value fluctuates but still maintains a downward trend.

6.3 Consensus Mechanism

In order to test the availability of this mechanism, we simulate a client RSU to send 10,000 transactions to conduct a pressure test on the blockchain.

As can be seen from Fig. 5(a) and 5(b), the average latency and throughput with our proposed consensus mechanism are significantly better than the PBFT consensus mechanism. Since our proposed consensus mechanism reduces the number of nodes participating in the block validation according to Algorithm 2, the throughput and average latency are greatly improved. Similarly, due to the selection of validator nodes based on stake, blockchain has similar throughput and average latency within a certain number of nodes.

According to the selection method of validator nodes from Algorithm 2, the parameter ξ 's influence on the latency change of the blockchain under different numbers of RSU nodes is tested. As shown in Fig. 5(c), the latency is proportional to the ξ 's value. When the ξ 's value is greater than or equal to 6, the latency gap between 30 and 60 nodes is smaller and stable.

7 Conclusion

In this paper, we propose a scalable blockchain-based trust management strategy for vehicular networks. Vehicles evaluate the credibility of the messages sent by surrounding vehicles through V2V communication according to objective factors and upload the results to a nearby RSU. Then, the RSUs collect the results and calculate the message senders' trust value. Utilizing blockchain, multiple RSUs can maintain a consistently distributed ledger that records vehicles' trust value changes. Through the analysis of security properties, simulation of trust evaluation mechanism and performance analysis of the blockchain, it is proved that this strategy has better availability and reliability. In addition, our proposed PoS and PBFT consensus mechanism improves the scalability and efficiency of the blockchain. The experiment result shows the strategy can provide a reliable and efficient decentralized trust management solution in an intelligent transportation environment.

Acknowledgements. This work was supported by Special Project for Research and Development in Key areas of Guangdong Province China (Grant No. 2020B0101090003).

References

1. Azees, M., Vijayakumar, P., Deborah, L.J.: Comprehensive survey on security services in vehicular ad-hoc networks. *IET Intel. Transport Syst.* **10**(6), 379–388 (2016)
2. Mühlbauer, R., Kleinschmidt, J.H.: Bring your own reputation: a feasible trust system for vehicular ad hoc networks. *J. Sens. Actuator Netw.* **7**(3), 37 (2018)
3. Lu, Z., Qu, G., Liu, Z.: A survey on recent advances in vehicular network security, trust, and privacy. *IEEE Trans. Intell. Transp. Syst.* **20**(2), 760–776 (2018)
4. Yan, Z., Zhang, P., Vasilakos, A.V.: A survey on trust management for Internet of Things. *J. Netw. Comput. Appl.* **42**, 120–134 (2014)
5. Yang, Z., Yang, K., Lei, L., Zheng, K., Leung, V.C.: Blockchain-based decentralized trust management in vehicular networks. *IEEE Internet Things J.* **6**(2), 1495–1505 (2018)
6. Baza, M., et al.: Detecting Sybil attacks using proofs of work and location in VANETs. *IEEE Trans. Dependable Secure Comput.* **19**(1), 39–53 (2020)
7. Hasrouny, H., Samhat, A.E., Bassil, C., Laouiti, A.: Trust model for secure group leader-based communications in VANET. *Wireless Netw.* **25**(8), 4639–4661 (2019). <https://doi.org/10.1007/s11276-018-1756-6>
8. Lahbib, A., Toumi, K., Laouiti, A., Laube, A., Martin, S.: Blockchain based trust management mechanism for IoT. In: 2019 IEEE Wireless Communications and Networking Conference (WCNC), pp. 1–8. IEEE (2019)
9. Javaid, U., Aman, M.N., Sikdar, B.: A scalable protocol for driving trust management in internet of vehicles with blockchain. *IEEE Internet Things J.* **7**(12), 11815–11829 (2020)
10. Kudva, S., Badsha, S., Sengupta, S., La, H., Khalil, I., Atiquzzaman, M.: A scalable blockchain based trust management in VANET routing protocol. *J. Parallel Distrib. Comput.* **152**, 144–156 (2021)
11. Raya, M., Papadimitratos, P., Gligor, V.D., Hubaux, J.P.: On data-centric trust establishment in ephemeral ad hoc networks. In: IEEE INFOCOM 2008-The 27th Conference on Computer Communications, pp. 1238–1246. IEEE (2008)
12. Kouicem, D.E., Imine, Y., Bouabdallah, A., Lakhlef, H.: A decentralized blockchain-based trust management protocol for the Internet of Things. *IEEE Trans. Dependable Secure Comput.* **19**(2), 1292–1306 (2022)
13. Kiayias, A., Russell, A., David, B., Oliynykov, R.: Ouroboros: a provably secure proof-of-stake blockchain protocol. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10401, pp. 357–388. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_12



BP-CODS: Blind-Spot-Prediction-Assisted Multi-Vehicle Collaborative Data Scheduling

Tailai Li, Chaokun Zhang^(✉), and Xiaobo Zhou

College of Intelligence and Computing, Tianjin University, Tianjin 300350, China
{litailai,zhangchaokun,xiaobo.zhou}@tju.edu.cn

Abstract. The most important thing for Connected and Automated Vehicles (CAVs) is to ensure driving safety and prevent the loss of life and property due to danger. The existence of vehicle blind spots can lead to incomplete or ineffective access to information, which will bring risks. At the same time, the transmission of a large amount of duplicate data will lead to information redundancy and bandwidth waste. In this paper, we design BP-CODS, which uses blind-spot prediction assistance to schedule image data between vehicles with the support of the Edge Server. We model the data scheduling transmission as two processes of uploading and downloading, form the set coverage problem, and propose a heuristic algorithm to solve it. We conduct extensive simulation experiments in CARLA to verify the effectiveness of BP-CODS in reducing a large number of redundant data.

Keywords: Scheduling · Vehicular networks · Prediction · Edge server

1 Introduction

In our daily life, the blind spot of the driver can lead to the operation mistake of the vehicle and may cause a serious accident. According to World Health Organization (WHO) report survey in 2018, the number of road traffic deaths reached 1.35 million every year [11]. Road accidents due to blind spots account for approximately 20% of overall lane-changing accidents [6]. Connected and Autonomous Vehicles (CAVs) were created to enable a safer transportation environment. Cellular Vehicle to Everything (C-V2X) based on 5G communication technology can realize data transmission, the sharing of data and computing resources [3]. Through vehicle cooperative communication, vehicles can obtain traffic conditions from their blind spots, and perceive potentially hazardous situations over an extended spatiotemporal range.

This work is supported in part by National Key R&D Program of China under Grant 2019YFB2102400, in part by National Natural Science Foundation of China under Grant No. 62072330, in part by China Postdoctoral Science Foundation 2020M680906, in part by Hebei Province High-level Talent Funding Project B202003027, in part by Tianjin Research Innovation Project for Postgraduate Students 2021YJSO2S04.

Through collaboration and sharing between CAVs, vehicles can receive a large amount of useful data information. For large amounts of data transmission, previous work focused more on low latency and high speed of vehicle data transmission, aiming at timely and stable transmission, with little attention to the data content. This will lead to the following problems. First, the data transmission is started on time, but the high mobility of the vehicle may result in untimely data reception. Second, in the transmission, the data of each vehicle is treated equally, but the importance of each data is different for each vehicle, and the more important data is not given priority in the transmission. Third, different vehicles may get the same or similar data, and their duplicate transmissions will also cause data redundancy. The transmission of such a large amount of redundant data will take up substantial bandwidth resources, leading to untimely transmission and failure to prioritize data, resulting in irreversible consequences.

In this paper, we focus on the image data obscured by the blind spot, aiming to supplement the obscured blind-spot view. The aid of blind-spot prediction allows the data to be transmitted in advance, enabling the vehicle to receive the data in time. The data acquired through multi-vehicle collaboration will be selected at the Edge Server for content selection and priority transmission of important data, which will reduce a large amount of redundant data, reduce the load on the network, and better ensure effective data acquisition. The main contributions of this paper are as follows.

- We propose BP-CODS for collaborative vehicle data scheduling aided by blind-spot prediction with the Edge Server. We exploit the sociality among vehicles using a graph neural network to perform blind-spot prediction.
- We upload the vehicle data to the Edge Server, select the content and send it down to the desired vehicles. We model this processing to minimize the amount of data downloaded. It is a set coverage problem belonging to NP-Hard. We hence propose heuristic algorithms to solve it.
- We validate BP-CODS in the CARLA simulator, and experiments show that our algorithm can substantially reduce the transmission of the same or similar data, reducing data redundancy and bandwidth waste.

2 Related Work

Vehicle collaboration is primarily designed to expand the perception boundary of vehicles and share information between vehicles to build safer road traffic environments. Among them, LiveMap [9] uses crowdsourced data from connected vehicles to detect, match, and track objects on the road and effectively integrate objects from multiple vehicles. EdgeSharing [8] leverages the edge cloud platform and real-time 3D feature maps to provide accurate localization and object sharing services for vehicles passing through intersections. EMP [17] can share raw sensor data between vehicles and use Edge Servers to merge individual views of vehicles to form a more complete view with higher resolution, greatly improving the quality of vehicle perception.

Data transmission in in-vehicle networks is mainly divided into vehicle-to-vehicle and vehicle-to-infrastructure data sharing and transmission. For vehicle-to-vehicle mode, D. Tian et al. [13] developed a robust optimization model for distributing content data traffic among different collaborative transmission paths to reduce unsuccessful transmissions. V2V-CoVAD [14], a vehicle-to-vehicle collaborative video alert propagation mechanism, designs a bidirectional collaborative transmission strategy for the transmission of accident videos in highway scenarios. For vehicle-to-infrastructure mode, after prefetching the contents of interest to RSU, S. Berri et al. [2] used different physical layer data rates to broadcast different contents to vehicles, and under the limitation of the RSU storage capacity, proposed a heuristic content prefetching and scheduling algorithms to maximize the broadcast downlink data contents. In [7], it formulates a hybrid scheduling problem, aiming to better exploit the synergy between centralized scheduling within RSU and self-organized data scheduling outside RSU, and uses an RSU Cooperation-based Adaptive Scheduling (RCAS) algorithm to solve the problem.

3 Model and Problem Formulation

3.1 Framework Overview

We consider a scenario with multiple vehicles driving on the road, each connected to an Edge Server (ES). The overall architecture of BP-CODS is shown in Fig. 1. Vehicle V_i captures forward information via cameras, uses the target detection and tracking algorithm to detect and track the targets, and then performs blind-spot prediction to obtain the movement orientation and the variation of the blind spot in the future. Vehicle V_i sends a data scheduling request to ES and then ES executes the uploading and downloading scheduling according to our proposed algorithm.

3.2 Model

Let N be the set of vehicles driving along the segment of the road where ES is implemented. The vehicles can send data scheduling requests and data to ES or receive vehicle data from ES. A vehicle that requires data from the other vehicle is denoted V_i , and the other that provides data to vehicle V_i is denoted V_j , where $i, j \in N$. We divide the time the vehicle is connected to ES into equal time block t^B . Let T and L be the sets of the time blocks and the transmission rate, respectively. At each time block $t \in T$, data is transmitted between the vehicle and ES at a given transmission rate $l, l \in L$, and its transmission time is denoted $o_l, l \in L$. Hence, here we define the ratio of time occupied by the transmission of the vehicle data at rate l as $r_l = \frac{o_l}{t^B}, l \in L$.

In the uploading process, we define the binary variable $k_i, i \in N$ to indicate whether vehicle V_i needs data scheduling, and the binary variable $z_{i,j}^+, i, j \in N$ to indicate whether vehicle V_i needs vehicle V_j to upload its data to ES. We

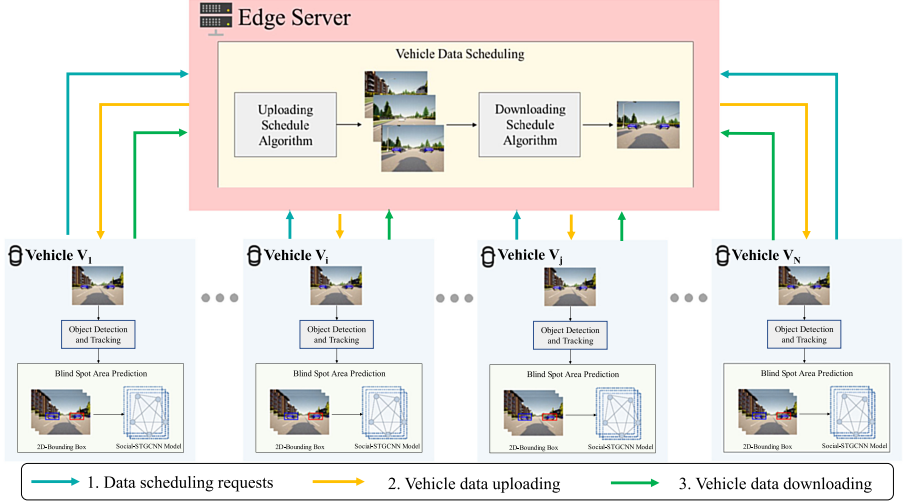


Fig. 1. The architecture of our proposed BP-CODS.

also define the binary variable $x_{j,l,t}^+, j \in N, l \in L, t \in T$ to indicate whether the data of vehicle V_j is uploaded to ES at transmission rate l in time block t . Let $\gamma_{j,l,t}^+, j \in N, l \in L, t \in T$ denote the probability of successful reception of the data uploaded to ES at transmission rate l within time block t . Note that, $\gamma_{j,l,t}^+$ and $\gamma_{j,l,t}^-$ can be obtained by a function of distance and data rate, which is described in detail in the experimental setup. Considering whether the data rate can be adaptively selected in the data transmission process, two modes are given here, the union model and the single model, to distinguish. Furthermore, let $u_{i,j,t}^+, i, j \in N, t \in T$ denote the reception rate of vehicle V_j 's data by ES due to the request of vehicle V_i at time block t . From the definition, $u_{i,j,t}^+ = \sum_{l \in L} \gamma_{j,l,t}^+ x_{j,l,t}^+ z_{i,j}^+, \forall i, j \in N, t \in T$. In addition, let $\delta_{i,j}$ be the distance between vehicle V_i and vehicle V_j , and dis be the distance threshold to determine whether the collaboration request are allowed.

Again, in the downloading process, we define the variables $z_{i,j}^-, x_{j,l,t}^-, \gamma_{j,l,t}^-, u_{i,j,t}^-$ accordingly. In particular, $u_{i,j,t}^- = \sum_{l \in L} \gamma_{j,l,t}^- x_{j,l,t}^- z_{i,j}^-, \forall i, j \in N, t \in T$. In each time block $t \in T$, we define set E_t to represent all objects detected by all vehicles. We hence define a binary variable C_{j,e_m} to represent whether the vehicle V_j detects object $e_m, e_m \in E_t = \{e_1, \dots, e_m\}$.

3.3 Problem Formulation

Uploading Process. Constraint (1) guarantees that only one data rate can be used to upload data to ES within a time block per vehicle. Constraint (2) limits the proportion of data transmission time between the vehicle and ES. Constraints (3) - (5) are binary variables.

Downloading Process. Constraint (6) guarantees that every object $e_m \in E_t$ must be detected by at least one vehicle. Constraint (7) indicates that the downloaded data must be chosen from uploaded data in the same time block. The meaning of Constraints (8) - (11) are similar to that of the uploading process.

Upload:

$$\sum_{l \in L} x_{j,l,t}^+ \leq 1, \forall j \in N, \forall t \in T \quad (1)$$

$$\sum_{j \in N, l \in L} x_{j,l,t}^+ r_l \leq 1, \forall t \in T \quad (2)$$

$$x_{j,l,t}^+ \in \{0, 1\}, \forall j \in N, \forall l \in L, \forall t \in T \quad (3)$$

$$z_{i,j}^+ = \begin{cases} 1, & \text{if } \delta_{i,j} \leq \text{dis} \ \& \ k_i = 1 \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

$$k_i \in \{0, 1\}, \forall i \in N \quad (5)$$

Download:

$$\sum_{j \in N} C_{j,e_m} u_{i,j,t}^- \geq 1, \quad (6)$$

$$\forall e_m \in E_t, \forall i \in N, \forall t \in T \quad (6)$$

$$u_{i,j,t}^- \leq u_{i,j,t}^+, \forall i, j \in N, \forall t \in T \quad (7)$$

$$\sum_{l \in L} x_{j,l,t}^- \leq 1, \forall j \in N, \forall t \in T \quad (8)$$

$$\sum_{j \in N, l \in L} x_{j,l,t}^- r_l \leq 1, \forall t \in T \quad (9)$$

$$x_{j,l,t}^- \in \{0, 1\}, \forall j \in N, \forall l \in L, \forall t \in T \quad (10)$$

$$z_{i,j}^- \in \{0, 1\}, \forall i, j \in N \quad (11)$$

Objective Function. For a given time $t = t_0$ (t_0 is the current time), we aim to design a vehicle data uploading and downloading scheme at ES that minimizes the amount of vehicle V_j 's data received by the vehicle V_i from ES in every determined time block. Meanwhile vehicle V_j 's data contains as many targets as possible. It is a set coverage problem, and the set coverage problem belongs to NP-Hard.

$$\min \sum_{i \in N} \sum_{j \in N} u_{i,j,t}^- \quad (12)$$

4 Algorithm Design

4.1 Blind-Spot Prediction Module

We now describe the blind-spot prediction module. It is not appropriate to schedule vehicle data too early or too late. We believe that it is a more reasonable and accurate method to decide the data scheduling time according to the future trend of blind spot size.

Let H be the set of blind spots. At time t , the 2D bounding box of blind spot h is defined as $X_t^h = \{(x_{1,t}^h, y_{1,t}^h), (x_{2,t}^h, y_{2,t}^h) \mid t \in T, h \in H$. We put X_t^h into the Social-STGCNN model [10] to predict the future coordinates denoted as $\hat{X}_{t+1}^h = \{(\hat{x}_{1,t+1}^h, \hat{y}_{1,t+1}^h), (\hat{x}_{2,t+1}^h, \hat{y}_{2,t+1}^h)\}$. Here, Social-STGCNN is a graph neural network model, which mainly consists of Spatio-Temporal Graph Convolution Neural Network (ST-GCNN) and Time-Extrapolator Convolution Neural Network (TXP-CNN). It can model social interactions and extract features for prediction. Then, we calculate the pixel size of the 2D bounding box $\hat{p}_{t+1}^h, t \in T, h \in H$ based on the predicted coordinates of the blind spots \hat{X}_{t+1}^h .

If there is \hat{p}_{t+1}^h greater than a certain threshold P , set $k_i = 1$ for vehicle V_i corresponding to blind spot h , 0 otherwise.

For the training of the Social-STGCNN model, we assume that $(x_{1,t}^h, y_{1,t}^h)$ follows the binary Gaussian distribution $N(\mu_{1,t}^h, \sigma_{1,t}^h, \rho_{1,t}^h)$, where $\mu_{1,t}^h$ is the mean of the distribution, $\sigma_{1,t}^h$ is the variance, and $\rho_{1,t}^h$ is the correlation [10] [4]. The assumption of $(x_{2,t}^h, y_{2,t}^h)$ can be given accordingly. Finally, the model parameters are obtained by minimizing the negative log-likelihood loss, that is, $L^h(W) = -\sum_{t \in T} \log P(x_{1,t}^h, y_{1,t}^h | \mu_{1,t}^h, \sigma_{1,t}^h, \rho_{1,t}^h) - \sum_{t \in T} \log P(x_{2,t}^h, y_{2,t}^h | \mu_{2,t}^h, \sigma_{2,t}^h, \rho_{2,t}^h)$, where W includes all trainable parameters of the model.

4.2 Vehicle Data Scheduling

Data scheduling is divided into two stages, upload and download, and we give the corresponding heuristic algorithms in sequence.

Uploading Process. The uploading scheduling process determines the set of vehicle V_j data to be uploaded, and it is summarized in Algorithm 1. The detailed process is executed in every time block according to the following steps.

First, we iterate over all vehicles. Based on $\delta_{i,j}$ and the scheduling instructions k_i for each vehicle, we assign a value to $z_{i,j}^+$. We calculate the minimum D_j , which is the ratio of the distance between vehicle V_i and vehicle V_j to the distance threshold dis .

Second, we calculate $s_{j,l,t}$ which refers to the number of all vehicles V_i that require vehicle V_j 's data at time block t with data rate l . We calculate $\omega_{j,l,t}$, which is the ratio of the number of vehicles V_i that need data to the occupied time by transmitting the data and occupied distance. This indicates whether scheduling this data at time block t with data rate l has a high performance-cost ratio.

Next, we sort $\omega_{j,l,t}$ into decreasing order. According to $\omega_{j,l,t}$, we search the pair of (j, l) in turn and obtain the scheduling order if (1) it does not violate the bandwidth constraint in (2); (2) the number of vehicles V_i that request vehicle V_j 's data (defined as B_j) is not 0 and the data of vehicle V_j has not been allocated to be uploaded to ES in this time block yet. Once its scheduling order is determined, τ , B_j and $x_{j,l,t}^+$ are updated.

Finally, we calculate $u_{i,j,t}^+$ and output its set $U_t^+ = \{u_{i,j,t}^+, \forall i, j \in N$.

Downloading Process. The downloading scheduling algorithm determines the set of data to be downloaded to vehicle V_i from ES, and it is summarized in Algorithm 2. The detailed process is executed in every time block according to the following steps.

First, for each vehicle V_i , a set F_i is constructed, to which vehicle V_j 's data that has been uploaded to ES and required by vehicle V_i is added.

Next, for each $u_{i,j,t}^+ \in F_i$, we calculate $count_j$, which refers to the number of objects in every data of vehicle V_j . We then sort $count_j$ into decreasing order,

and obtain the order of its subscript j . Iterating through j , if has objects in the vehicle V_j 's data that are not selected, it will set all objects of vehicle V_j to the selected state, and set $z_{i,j}^- = 1$, indicating that vehicle V_j 's data is sent to vehicle V_i from ES. Mimicking steps 9-21 of the uploading scheduling algorithm, we can get $x_{j,l,t}^-$.

Finally, $u_{i,j,t}^-$ is calculated and $U_t^- = \{u_{i,j,t}^-\}, \forall i, j \in N$ is output.

5 Experiment

5.1 Setting

Environment. We adopt the urban driving simulator CARLA [5], an open-source simulator implemented in Unreal Engine 4, to validate BP-CODS, and OpenCDA [15] to implement the simulated vehicles for autonomous driving. The

Algorithm 1: Uploading Scheduling Algorithm for each time block t

Input: N, L, Γ^+, K

Output: U_t^+

```

1 for each  $i \in N$  do
2   if  $k_i == 1$  then
3     for each  $j \in N$  do
4       if  $\delta_{i,j} \leq dis$  and  $i \neq j$  then
5          $z_{i,j}^+ = 1$ ;
6          $D_j = \min(\frac{\delta_{i,j}}{dis})$ ;
7       else
8          $z_{i,j}^+ = 0$ ;
9 for each  $j \in N$  and  $l \in L$  do
10   $s_{j,l,t} = \sum_{i \in N} \gamma_{j,l,t}^+ z_{i,j}^+$ ;
11   $\omega_{j,l,t} = \frac{s_{j,l,t}}{r_l D_j}$ ;
12 Sort  $\omega_{j,l,t}$  into decreasing order;
13 Let  $A$  represent the subscript sequence pair  $(j, l)$  of  $\omega_{j,l,t}$ ;
14 Set  $B_j = \sum_{i \in N} z_{i,j}^+, x_{j,l,t}^+ = 0, \tau = 0$ ;
15 for  $(j, l) \in A$  do
16   if  $\tau > 1$  then
17     Break;
18   if  $B_j > 0$  and  $\sum_{l \in L} x_{j,l,t}^+ \leq 0$  then
19      $B_j = \max(0, B_j - s_{j,l,t})$ ;
20      $x_{j,l,t}^+ = 1$ ;
21      $\tau + = r_l$ ;
22  $u_{i,j,t}^+ = \sum_{l \in L} \gamma_{j,l,t}^+ x_{j,l,t}^+ z_{i,j}^+$ ;
23 return  $U_t^+ = \{u_{i,j,t}^+\}, \forall i, j \in N$ ;

```

hardware and software we used consist of Intel Xeon Silver 4208, NVIDIA RTX 3090, and Ubuntu 20.04 LTS.

The target detection and tracking module in BP-CODS can be implemented by various target detection and tracking algorithms such as YOLOv5, SSD, mono3DT, PCA. Here we obtain target classes and 2D bounding boxes for vehicle detection directly in the CARLA world, and then we can get E_t and C_{j,e_m} .

Datasets. In the training of the blind-spot prediction module, we used the BDD100K dataset [16], an autonomous driving dataset released in 2018. We extracted the positions of 2D bounding boxes for bus, car, and truck classes, and acquired continuous sequence data over 10,000 frames. We set the training batch size to 128 and trained the model for 500 epochs using stochastic gradient descent (SGD). The initial learning rate is 0.01, which becomes 0.002 after 150 epochs.

Simulation Settings. We used the town06 map in the CARLA simulator and we chose a 5-lane straight road close to 700 m long. We distribute 5–20 Connected and Automated Vehicles (CAVs) equipped with BP-CODS on the road. We tested two scenarios. In Scenario A, the speed of all CAVs is set to 30 km/h, while in Scenario B, the speed of each CAV is set to 30 km/h–70 km/h respectively.

Algorithm 2: Downloading Scheduling Algorithm for each time block t

Input: $N, L, \Gamma^-, U_t^+, C_{j,e_m}$;
Output: U_t^- ;

- 1 **for** each $i \in N$ **do**
- 2 Set $F_i = \emptyset$;
- 3 **for** each j **in** N **do**
- 4 **if** $u_{i,j,t}^+ == 1$ **then**
- 5 $F_i = F_i \cup u_{i,j,t}^+$;
- 6 **for** each $u_{i,j,t}^+ \in F_i$ **do**
- 7 $count_j = \sum_{e_m \in E_t} C_{j,e_m} u_{i,j,t}^+$;
- 8 Sort $count_j$ into decreasing order;
- 9 Let R represent the subscript sequence j of $count_j$;
- 10 Initialize E_t ;
- 11 **for** each $j \in R$ **do**
- 12 **if** $\sum_{e_m \in E_t} C_{j,e_m} e_m < count_j$ **then**
- 13 $z_{i,j}^- = 1$;
- 14 Set all objects $e_m = 1$ in the data of the vehicle V_j ;
- 15 Compute $x_{j,l,t}^-$ imitating steps 9-21 of Algorithm 1;
- 16 $u_{i,j,t}^- = \sum_{l \in L} \gamma_{j,l,t}^- x_{j,l,t}^- z_{i,j}^-$;
- 17 **return** $U_t^- = \{u_{i,j,t}^-, \forall i, j \in N\}$;

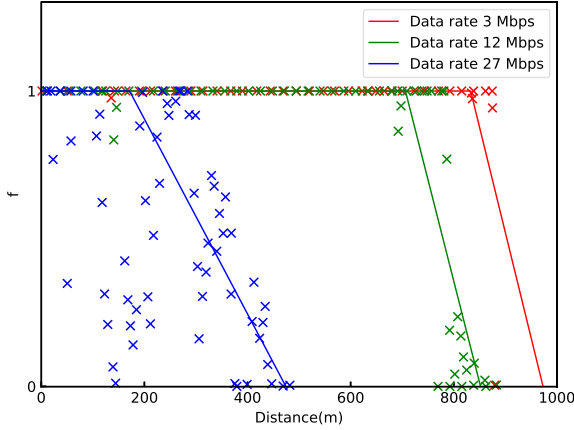


Fig. 2. The data reception probability versus distance between vehicle and ES at different data rates.

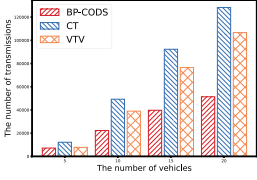
We place ES at 250 m behind the starting point of the road so that the farthest distance between the vehicle and ES during driving is close to 1000 m. We perform data synthesis based on the data from the experimental result in [1, 12] and obtain the synthesized data of the relationship between the data reception rate and the distance between the vehicle and ES under the 5G protocol, as shown in Fig. 2.

We call the mode that can transmit data at multiple rates in combination as the union model, and the mode that can only transmit at a single rate as the single model. In the simulation, we use a function that randomly generates probabilities to simulate the process of receiving or discarding image data probabilistically during transmission.

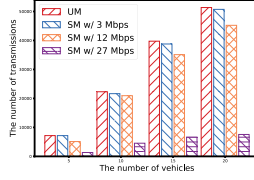
We set time block t to 1 s, distance threshold dis to 50 m, and blind spot threshold P to 526. Each CAV is mounted with a front-facing camera with a maximum sensing range of 50 m. In the CARLA world, the size of the image data acquired by each camera is 28800 B. We give the set of available transmission rates $L = \{3 \text{ Mbps}, 12 \text{ Mbps}, 27 \text{ Mbps}\}$. Hence the time to transmit each image is 0.0732 s, 0.0183 s, and 0.0081 s, respectively.

Baselines. We have implemented two baselines, namely *CT* and *VTV* to compare BP-CODS.

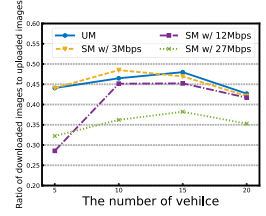
- *CT*, an algorithm in [2], can upload data to ES and then send it to the vehicle uniformly. We modify it to fit the CARLA simulation environment.
- *VTV*, another algorithm in [13], can send the data of the front vehicle directly to the rear vehicle without going through ES. We modify it to fit the CARLA simulation environment.



(a) Comparison of transmission number

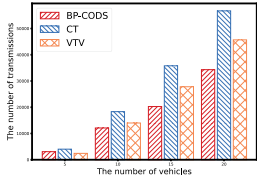


(b) Impact of the transmission model in BP-CODS

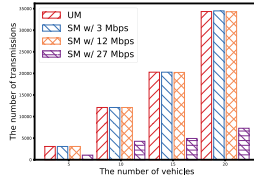


(c) Reduction ratio of image data in BP-CODS

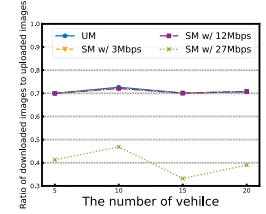
Fig. 3. Experiments in Scenario A. We use UM and SM refer to the union model and the single model.



(a) Comparison of transmission number



(b) Impact of the transmission model in BP-CODS



(c) Reduction ratio of image data in BP-CODS

Fig. 4. Experiments in Scenario B.

5.2 Performance Analysis

We set the simulated vehicles to be 5, 10, 15, and 20 CAVs, respectively. Due to the different speeds of CAVs in the two scenarios, their scheduling on the same road is also different. Scenario A is scheduled 1200 times while Scenario B is scheduled 600 times. In Fig. 3(a) and Fig. 4(a), we compare BP-CODS with the other algorithms CT and VTV under the union model in Scenario A and Scenario B. We find that BP-CODS reduces the data transmission number by 10%–50% in Scenario A and reduces the data transmission number by up to 20% in Scenario B, compared to VTV. Compared with CT, the transmission number of BP-CODS in Scenario A is reduced by 40%–60% and the data transmission number in Scenario B is reduced by 20%–40%. It is foreseeable that as the number of CAV equipped with BP-CODS increases, we can reduce the number of data transmissions even more, while still maintaining the validity and criticality of the transmitted data.

Next, we investigate the performance of BP-CODS in different transmission models (i.e., union model and single model). In Fig. 3(b) and Fig. 4(b), we compare the number of transmissions of BP-CODS for different transmission models in Scenario A and Scenario B. We can find that the union model has more stable performance, which can take into account both network data rate and data reception rate. When using a single rate of 27 Mbps and 12 Mbps, its communi-

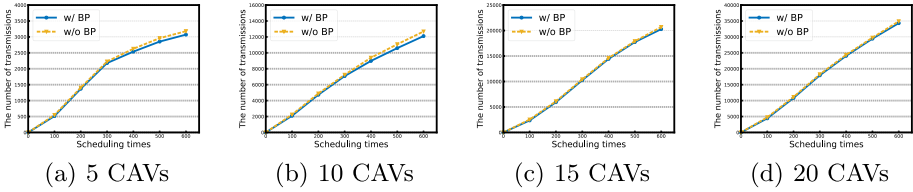


Fig. 5. The number of transmissions with and without the blind-spot prediction. We use BP refer to the Blind-spot Prediction Module.

cation distance is short, and the data reception rate decreases after exceeding the communication range, and the number of transmissions decreases significantly. While using the 3 Mbps rate will have great transmission stability, but the transmission rate is also the lowest, which can't use the bandwidth resources reasonably. Therefore, the union model can achieve a good balance between data rate and transmission stability.

We believe that if both CAVs have the same object in their captured image data at the same moment, the two images are duplicates and only one of them needs to be transmitted. In Fig. 3(c) and Fig. 4(c), we compare the ratio of the number of images downloaded to the number of uploaded images for BP-CODS under different transmission models and different number of CAVs in Scenario A and Scenario B. We find that BP-CODS can reduce the image data by more than 50% in Scenario A and the image data more than 30% in Scenario B. Hence, our algorithm can reduce the duplicate transmission of large amounts of image data.

Finally, we conduct ablation experiments to investigate the effect of the blind-spot prediction module. Since the CAVs in Scenario A are all at the same speed, which does not show the effect of the blind-spot prediction module very well, we only select Scenario B to carry out these experiments. In Fig. 5, we compare the situation with and without the blind-spot prediction module. We find that for different numbers of CAVs, the difference between the number of transmissions with and without the blind-spot prediction module increases as the number of scheduling increases. The inclusion of the blind-spot prediction module can reduce data transmission by 2%–5%.

6 Conclusion

In this paper, we introduce BP-CODS, the overall system architecture of multi-vehicle collaborative vehicle data scheduling assisted by blind-spot prediction using the graph neural network. Using a union model for data transmission in vehicle data scheduling makes more rational use of bandwidth resources and also reduces the number of data transmissions. Our simulation in CARLA shows that it can reduce the number of data transmission by more than 60%, and it can reduce the transmission of similar or duplicate data by more than 30%.

References

1. Association, G.A., et al.: An assessment of lte-v2x (pc5) and 802.11 p direct communications technologies for improved road safety in the eu. 5G Automotive Association, Technical Report, December (2017)
2. Berri, S., Zhang, J., Bensaou, B., Labiod, H.: Content-prefetching and broadcast scheduling in vehicular networks with a realistic channel model. In: IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 1–7. IEEE (2019)
3. Chen, S., Hu, J., Shi, Y., Zhao, L., Li, W.: A vision of c-v2x: technologies, field testing, and challenges with Chinese development. *IEEE Internet Things J.* **7**(5), 3872–3881 (2020)
4. Deo, N., Trivedi, M.M.: Convolutional social pooling for vehicle trajectory prediction. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 1468–1476 (2018)
5. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: CARLA: An open urban driving simulator. In: Conference on robot learning, pp. 1–16. PMLR (2017)
6. Farmer, C.M.: Crash avoidance potential of five vehicle technologies. Insurance Institute for Highway Safety (2008)
7. Ko, B., Liu, K., Son, S.H., Park, K.J.: RSU-assisted adaptive scheduling for vehicle-to-vehicle data sharing in bidirectional road scenarios. *IEEE Trans. Intell. Transp. Syst.* **22**(2), 977–989 (2020)
8. Liu, L., Gruteser, M.: EdgeSharing: edge assisted real-time localization and object sharing in urban streets. In: IEEE INFOCOM 2021-IEEE Conference on Computer Communications, pp. 1–10. IEEE (2021)
9. Liu, Q., Han, T., Xie, J.L., Kim, B.: LiveMap: real-time dynamic map in automotive edge computing. In: IEEE INFOCOM 2021-IEEE Conference on Computer Communications, pp. 1–10. IEEE (2021)
10. Mohamed, A., Qian, K., Elhoseiny, M., Claudel, C.: Social-STGCNN: a social spatio-temporal graph convolutional neural network for human trajectory prediction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 14424–14432 (2020)
11. Organization, W.H., et al.: Global status report on road safety 2018: summary. World Health Organization, Technical Report (2018)
12. Shagdar, O., Tsukada, M., Kakiuchi, M., Toukabri, T., Ernst, T.: Experimentation towards ipv6 over ieee 802.11 p with its station architecture. In: International Workshop on IPv6-based Vehicular Networks (colocated with IEEE Intelligent Vehicles Symposium) (2012)
13. Tian, D., Zhou, J., Chen, M., Sheng, Z., Ni, Q., Leung, V.C.: Cooperative content transmission for vehicular ad hoc networks using robust optimization. In: IEEE INFOCOM 2018-IEEE Conference on Computer Communications, pp. 90–98. IEEE (2018)
14. Wang, S., Zhang, Q., Chen, G.: V2V-CoVAD: a vehicle-to-vehicle cooperative video alert dissemination mechanism for Internet of Vehicles in a highway environment. *Veh. Commun.* **33**, 100418 (2022)
15. Xu, R., Guo, Y., Han, X., Xia, X., Xiang, H., Ma, J.: OpenCDA: an open cooperative driving automation framework integrated with co-simulation. In: 2021 IEEE International Intelligent Transportation Systems Conference (ITSC), pp. 1155–1162. IEEE (2021)

16. Yu, F., et al.: BDD100K: A diverse driving video database with scalable annotation tooling. arXiv preprint [arXiv:1805.04687](https://arxiv.org/abs/1805.04687) **2**(5), 6 (2018)
17. Zhang, X., et al.: EMP: edge-assisted multi-vehicle perception. In: Proceedings of the 27th Annual International Conference on Mobile Computing and Networking, pp. 545–558 (2021)



Performance Analysis of Partition-Based Caching in Vehicular Networks

Siyuan Zhou^{1,2}(✉), Wei Wu², and Guoping Tan^{1,2}

¹ Key Laboratory of Water Big Data Technology of Ministry of Water Resources, Hohai University, Nanjing, China

siyuan.zhou@hhu.edu.cn gptan@hhu.edu.cn

² School of Computer and Information, Hohai University, Nanjing, China

wei.wu@hhu.edu.cn

Abstract. Partition-based caching is emerging as an appealing solution to improve the performance of the content caching by increasing the content diversity at the network edge. In this paper, we model and analyze a vehicular network where the vehicles can obtain the requested contents from the roadside units (RSUs) by adopting the random linear network coding in the partition-based caching scheme. Specifically, the geographic distribution of the roads and RSUs are modeled by the stochastic geometry tools. The required content can be obtained from the multiple nearest RSUs and the content can be decoded by using the successive interference cancellation approach. We derive the distance distribution between the typical vehicle and the nearest RSUs, and obtain the analytical expression of the successful transmission probability of the content caching. The numerical simulations verify the analytical results and provide the guidelines for the application of the partition-based caching in vehicular networks.

Keywords: Stochastic geometry · Content caching · Vehicular networks

1 Introduction

With the rapid growth of the vehicular networks, the explosive data demand from the vehicles typically requires the networks to support the data rates up to several Gbps from the data center to the vehicles. It is found that the nearby vehicles usually download the same up-to-date 3D high-resolution maps [1]. Additionally, vehicles tend to make content request based on the content popularity. Considering these facts, content caching at the edge of the vehicular networks has been proposed for the content delivery by reducing the communication overhead between the vehicles and the data center [2].

In the content caching architecture, the network performance highly depends on the adopted caching strategies. The proposed caching strategies can be categorized into two types: the entire content caching and the partition-based caching. Most existing works consider the scenarios where the entire contents are cached at the BSs or the

This work was supported in part by the National Natural Science Foundation of China (No. 61701168, 61832005), the China Postdoctoral Science Funded Project(No. 2019M651672).

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022

L. Wang et al. (Eds.): WASA 2022, LNCS 13473, pp. 309–318, 2022.

https://doi.org/10.1007/978-3-031-19211-1_26

RSUs [1, 3, 4]. In [3], the entire contents are cached at the network edge based on the contents' popularity which is subject to the Zipf distribution. The most popular caching and the probabilistic caching strategies are proposed. The authors in [1] considered the contents are cached at the vehicles with the uniform distribution. And all contents have an equal probability of being cached in the edge node. In [4], a hybrid caching strategy is proposed, where the contents are divided into two groups: the most popular sets and the less popular sets. Network nodes at different layers cache contents with different popularity. In the aforementioned content caching scheme, the user has to search for the targeted edge node that has stored the required content. However, due to the limited storage of the edge nodes, it is impossible to store all entire contents in each edge node, and thus the targeted edge node might be far from the user. This motivates the application of the partition-based caching in the network.

In the partition-based caching scheme, the entire file can be decomposed into multiple subfiles and stored in multiple network nodes in an uncoded [5] or coded manner [5–8]. For the uncoded strategy [5], the different pieces of subfiles are directly stored in the edge nodes. In this case, the original content can be recovered only if all subfiles are received by the user and this is a stringent requirement in the vehicular communication scenario. In [6], the authors investigated the partition-based coded caching in the cellular network where the subfiles are coded according to the Random Linear Network Coding (RLNC) approach. The user can recover the entire file by obtaining coded subfiles from several recent BSs, which is easier to be satisfied compared to the uncoded approach. The authors in [7, 8] considered the combination of partitioned and non-partitioned based caching scheme in Fog Radio Access Networks. In [7], the transmission delay and the energy efficiency are first analytically investigated. Then, a multi-objective optimization problem is formulated to obtain the optimal hybrid caching strategy by considering the trade-off between the delay and the energy efficiency. The successful transmission probability and the fractional offloaded traffic are analyzed in [8] and the corresponding multi-objective optimization problems are investigated. However, the impact of the partition-based caching in vehicular networks and how it can improve the caching performance are still unknown.

In this paper, we consider a cache-enabled vehicular network with partition-based caching scheme. Stochastic geometry tools are used to characterize the random distribution of the vehicles and the RSUs. We consider a RLNC-based caching design and adopt the successive interference cancellation (SIC) approach to decode multiple encoded subfiles for the content recovery. We derive the expression of the successful transmission probability with specific caching parameters in RLNC. The results reveal that there is a trade-off between the successful transmission probability and the size of the cache resources occupied.

2 System Model

2.1 Network Model

We consider the RSUs are deployed along the roads and the vehicles connect with multiple RSUs in order to receive multiple subfiles, as illustrated in Fig. 1. We assume the distribution of roads are modeled as a Poisson line process (PLP) Φ_l with density

λ_l . PLP is proposed in [1] to characterize the distribution of roads in vehicular networks. Upon the distribution of the roads, the RSUs located along the L -th road can be characterized by a one-dimensional homogeneous Poisson point process (PPP) Ψ_L with density λ_2 . Thus, the locations of RSUs form a Poisson line cox process (PLCP) that is denoted as $\Phi_v \equiv \cup_{L \in \Phi_l} \Psi_L$ [9]. For analytical simplicity, we consider the typical vehicle is located at the origin. In addition, we consider a typical line L_0 through the origin. The RSUs on the typical line are modeled as an independent one-dimensional PPP Ψ_{L_0} with density λ_2 . As such, the distribution of RSUs is the combination of two independent distributions of Φ_v and Ψ_{L_0} , expressed as $\Phi_2 = \Phi_v \cup \Psi_{L_0}$. As illustrated in Fig. 1, the RSUs are labeled in ascending order based on their distance to the typical vehicle.

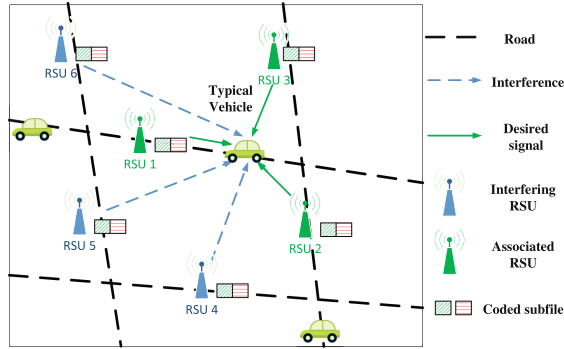


Fig. 1. System model of the partition-based caching in vehicular network with $N = 2$ and $m = 3$.

2.2 Partition-Based Caching Design

We assume there are N contents that can be requested by the vehicles and each content has the same size of S bits. Let $\mathbf{N} \triangleq \{1, 2, \dots, N\}$ represents the collection of N contents. The typical vehicle requests the content randomly based on the content popularity which is subject to the Zipf distribution. The probability of the typical vehicle requests a content $n \in \mathbf{N}$ is $p_n \in (0, 1)$, where $p_n = n^{-\gamma} / \sum_{n \in \mathbf{N}} n^{-\gamma}$.

Each content is decomposed into m subfiles at the data center. Then, all m subfiles are coded by using the Random linear network coding (RLNC) approach. We consider a large coding domain in RLNC, and the content can be fully recovered by any m coded subfiles [6]. At least one subfile of each content is distributed at every RSU during the off-peak hours. Thus, each RSU should at least have the storage capacity SN/m bits in order to cache the subfiles of all contents.

In this partition-based caching scheme with RLNC, all contents are cached in each RSU in the form of their coded subfiles. The typical vehicle sends the content request based on the content popularity. Once the content request is confirmed, the typical vehicle can select m RSUs for the content downloading without considering whether these m RSUs store the same subfiles. In order to improve the transmission success probability, we assume the typical vehicle is associated with the nearest m RSUs in this model.

2.3 Channel Model

We consider the downlink transmission between multiple RSUs and the typical vehicle. Each RSU has one antenna with the same transmission power P and transmission bandwidth W . The vehicle is also equipped with one antenna. We consider a discrete-time system where time is divided into multiple periods and one period is T seconds. The transmission of the subfile can be achieved in one period. As for the path loss, we assume that the transmitted signals with distance d are attenuated by a factor $d^{-\alpha}$, where $\alpha > 2$ is the path loss exponent. For small-scale fading we consider the Rayleigh fading as it can describe the multi-path propagation effect in the vehicular networks.

We consider an interference-limited transmission by ignoring the background thermal noise. It is assumed that the RSUs in the network are all active in the transmission periods. In this case, the RSUs that do not serve the typical vehicle are active, which incur the co-channel interference. Let d_i denote the distance between RSU i and the typical vehicle, where $d_1 \leq d_2 \leq \dots$. The received signals of the typical vehicle are: $y_n = \sum_{i \in \Phi_k} d_i^{-\frac{\alpha}{2}} h_i P + \sum_{j \in \Phi_2 \setminus \Phi_k} d_j^{-\frac{\alpha}{2}} h_j P$, where Φ_2 is the set of RSUs in the network, $\Phi_k \in \{1, 2, \dots, m\}$ refers to the set of the associated RSUs and h_i represents the small-scale channel fading between the i -th RSU and the typical vehicle.

When y_n is received, the typical vehicle adopts SIC to decode the transmission signals from the signal combination. SIC decoding is performed in order of distance from near to far. In particular, when the vehicle decodes the signal from the i -th RSU, we assume that the signals of RSUs closer than the i -th RSU have been successfully decoded and eliminated. The signal-to-interference ratio of the signal from RSU i is defined as:

$$SIR_i = \frac{d_i^{-\alpha} |h_i|^2}{\sum_{j \in \Phi_2 \setminus \{1, 2, \dots, i\}} d_j^{-\alpha} |h_j|^2} \tag{1}$$

2.4 Performance Metric

The successful transmission probability of the required content is used as the performance metric of the this caching scheme. According to the above, the successful transmission probability of the requested content n equals to the joint successful transmission probability of the transmission with multiple RSUs, which is written as:

$$P_n(m) = \Pr \left[\bigcap_{i \in \Phi_k} W \log_2 (1 + SIR_i) > \frac{S}{mT} \right] \tag{2}$$

where SIR_i is given in (1). According to the total probability theorem, the successful transmission probability of the content caching is given by: $Q = \sum_{n \in \mathbf{N}} p_n P_n(m)$, where p_n is the file request probability that is subject to Zipf distribution.

3 Performance Analysis

In this section, we analyze the SIR-based successful transmission probability of the typical vehicle. The successful transmission probability is closely related to the distribution of the transmission distance since all nodes are randomly distributed. Thus, the analysis begins with the derivation of the distance distributions between the associated RSUs and the typical vehicle. Then, the conditional probability of successfully decoding the signal of the j -th RSU is derived considering the desired signals from the nearest $j - 1$ RSUs are obtained and eliminated from the signal combination. Finally, the successful transmission probability of the content caching is obtained.

3.1 Serving Distance Distributions

The distance distribution between the typical vehicle and the nearest m RSUs is required to characterize the randomness of SIR. It should be noticed that the RSUs might be located on the typical line or the other lines and the distance distribution in these two scenarios are different. Thus, we first express the distance distribution in each scenario. Since the RSUs on the typical line is subject to 1D homogeneous PPP distribution, the probability of the existence of N_1 RSUs within a distance r to the typical vehicle is given by: $\mathbb{P}(N_1 = n) = \exp(-2\lambda_2 r) \frac{(2\lambda_2 r)^n}{n!}$, where the above expression is obtained by acknowledging that the number of nodes in a finite area in PPP model is Poisson distributed. Considering the RSUs on the roads except for the typical road, the probability of the existence of N_2 RSUs in the area with a radius of r to the typical vehicle is given by [9]: $\mathbb{P}(N_2 = k) = \frac{(-\lambda_2)^k}{k!} \left[\frac{\partial^k}{\partial s^k} \mathcal{L}_T(s) \right]_{s=\lambda_2}$, where $\mathcal{L}_T(s) = \exp \left[-2\pi\lambda_l \int_0^r 1 - \exp \left(-2s\sqrt{r^2 - \rho^2} \right) d\rho \right]$ is the Laplace transform of the total chord length distribution and ρ is the distance from the origin to a line in the area.

Given the above expressions, the probability of the distance between the typical vehicle and the j -th closest RSU is larger than r equals to the probability of having at most $j - 1$ RSUs in the area with radius r , which is given by:

$$P(d_j > r) = P(N < j - 1) = \sum_{i=0}^{j-1} \sum_{k=0}^{j-1-i} P(N_1 = i) P(N_2 = k) \quad (3)$$

where d_j is the distance between the typical vehicle and j -th closest RSU and we denote $N = N_1 + N_2$. As such, the cumulative distribution function (CDF) of d_j is expressed as:

$$F_{d_j}(r) = 1 - P(d_j > r) = 1 - \sum_{i=0}^{j-1} \sum_{k=0}^{j-1-i} \exp(-2\lambda_2 r) \frac{(2\lambda_2 r)^i}{i!} \frac{(-\lambda_2)^k}{k!} \left[\frac{\partial^k}{\partial s^k} \mathcal{L}_T(s) \right]_{s=\lambda_2} \quad (4)$$

Additionally, the probability density function (PDF) of d_j is given by:

$$f_{d_j}(r) = \frac{dF_{d_j}(r)}{dr} \quad (5)$$

3.2 SIR-based Successful Transmission Probabilities

The SIR-based successful transmission probability is defined as the probability that the received SIR from all associated RSUs satisfy $W \log_2(1 + \text{SIR}) > \frac{S}{mT}$. The successful transmission probability of caching content n from the nearest i RSUs is derived as:

$$P_n(i) = \Pr \left[\bigcap_{j \in \{1, 2, \dots, i\}} W \log_2(1 + \text{SIR}_j) > \frac{S}{mT} \right] \stackrel{(a)}{\approx} \prod_{j=1}^i \Pr \left[W \log_2(1 + \text{SIR}_j) > \frac{S}{mT} \right], \quad (6)$$

where (a) follows from the fact that the decoding processes among multiple RSUs can be approximated as the independent events. This independent approximation is also used in the performance analysis of partition-based caching in the large-scale cellular networks [6]. The accuracy of this approximation will be verified by simulation later.

By considering the distance $d_j = r$ is known, the conditional probability of successfully decoding the signal of the j -th RSU is derived as

$$\begin{aligned} P_n(j|r) &= \Pr \left[W \log_2(1 + \text{SIR}_j) > \frac{S}{mT} | d_j = r \right] \\ &\stackrel{(a)}{=} \mathbb{E}_{I_j} \left[\Pr \left[|h_j|^2 > \left(2^{\frac{S}{mT}} - 1 \right) d_j^\alpha I_j | d_j = r \right] \right] \\ &\stackrel{(b)}{=} \mathbb{E}_{I_j} \left[\exp \left(- \left(2^{\frac{S}{mT}} - 1 \right) r^\alpha I_j \right) \right] \\ &\stackrel{\Delta}{=} \mathcal{L}_{I_j}(s, r) \Big|_{s = \left(2^{\frac{S}{mT}} - 1 \right) r^\alpha} \end{aligned} \quad (7)$$

where the step (a) is according to the expression of SIR_i given in (1), and the step (b) is according to the fact that $|h_i|^2$ follows an exponential distribution with mean 1.

The characterization of the interference I_j should consider the exclusion area and the network topology in vehicular networks. Since SIC is adopted in this system, we consider the exclusion area is centered at the typical vehicle with radius r and there are no interfering RSUs in it. Therefore, the interference received at the origin I_j can be divided into three parts: $I_j = I_{j20} + I_{j21} + I_{j22}$. In this expression, the interference from the RSUs located on the typical line passing through origin is represented as $I_{j20} = \sum_{k \in \Psi_{l_0} \setminus \{1, 2, \dots, j\}} d_k^{-\alpha} |h_k|^2$. The interference from RSUs located on the line that intersect with the exclusion area is denoted by $I_{j21} = \sum_{k \in \Phi_v \setminus \{1, 2, \dots, j\}, \rho < r} d_k^{-\alpha} |h_k|^2$. At last, $I_{j22} = \sum_{k \in \Phi_v \setminus \{1, 2, \dots, j\}, \rho > r} d_k^{-\alpha} |h_k|^2$ is used to represent the interference from the RSUs located on the lines that do not intersect with the exclusion area.

The Laplace transform of I_{j20} is derived as:

$$\begin{aligned} \mathcal{L}_{I_{j20}}(s, r) &= \mathbb{E} \left[\exp \left(-s \sum_{k \in \Psi_{L_0} \setminus \{1, 2, \dots, j\}} d_k^{-\alpha} |h_k|^2 \right) \right] \\ &\stackrel{(a)}{=} \mathbb{E} \left[\prod_{k \in \Psi_{L_0} \setminus \{1, 2, \dots, j\}} \frac{1}{1 + s d_k^{-\alpha}} \right] \\ &\stackrel{(b)}{=} \exp \left(-2\lambda_2 \int_r^\infty \left(1 - \frac{1}{1 + s x^{-\alpha}} \right) dx \right) \end{aligned} \quad (8)$$

where (a) is obtained since $|h_k|^2$ follows an exponential distribution with mean 1, and (b) follows from the probability generating function (PGFL) of the one-dimensional PPP. The Laplace transforms of I_{j21} is given by:

$$\begin{aligned} \mathcal{L}_{I_{j21}}(s, r) &= \mathbb{E} \left[\exp \left(-s \sum_{k \in \Phi_v \setminus \{1, 2, \dots, j\}, \rho < r} |h_k|^2 d_k^{-\alpha} \right) \right] \\ &\stackrel{(a)}{=} \mathbb{E}_{\Phi_l} \left[\prod_{L \in \Phi_l} \mathbb{E}_{\Psi_L} \left[\prod_{k \in \Psi_L} \frac{1}{1 + s d_k^{-\alpha}} \mid \Phi_l, \rho < r \right] \right] \\ &\stackrel{(b)}{=} \mathbb{E}_{\Phi_l} \left[\prod_{(\rho, \theta) \in \Phi_l} \exp \left(-2\lambda_2 \int_{\sqrt{r^2 - \rho^2}}^\infty \frac{s}{s + (\rho^2 + \quad)^{\frac{\alpha}{2}}} d \right) \right] \\ &\stackrel{(c)}{=} \exp \left\{ -2\pi\lambda_l \int_0^r \left[1 - \exp \left(-2\lambda_2 \int_{\sqrt{r^2 - \rho^2}}^\infty \frac{s}{s + (\rho^2 + \quad)^{\frac{\alpha}{2}}} d \right) \right] d\rho \right\} \end{aligned} \quad (9)$$

where (a) is obtained by taking the expectation of $|h_k|^2$ and conditioning on the line process Φ_l . By expressing the location of the RSUs in polar coordinates, (b) is obtained by using the PGFL of the 1D PPP Ψ_L on each line. In the last step, (c) follows from the PGFL of the 2D PPP in the representation space corresponding to the line process Φ_l , where $d_k = (\rho^2 + \quad)^{\frac{1}{2}}$, ρ is the distance between origin and other road in the disc, and \quad is the distance between the foot of the perpendicular and RSUs on the road in the disc. Similarly, we can obtain the Laplace transforms of I_{j22} which is given by:

$$\mathcal{L}_{I_{j22}}(s, r) = \exp \left\{ -2\pi\lambda_l \int_r^\infty \left[1 - \exp \left(-2\lambda_2 \int_0^\infty \frac{s}{s + (\rho^2 + \quad)^{\frac{\alpha}{2}}} d \right) \right] d\rho \right\} \quad (10)$$

Therefore, the conditional probability of successfully decoding the signal of the j -th RSU is given by:

$$P_n(j|r) = \mathcal{L}_{I_j}(s, r) \Big|_{s=\left(2\frac{s}{WmT} - 1\right)r^\alpha} = \mathcal{L}_{I_{j20}}(s, r) \mathcal{L}_{I_{j21}}(s, r) \mathcal{L}_{I_{j22}}(s, r) \Big|_{s=\left(2\frac{s}{WmT} - 1\right)r^\alpha} \quad (11)$$

The overall successful transmission probability of a content requested by the typical vehicle for a given design parameter m is given by:

$$Q = \sum_{n \in \mathbf{N}} p_n P_n(m) = \sum_{n \in \mathbf{N}} p_n \prod_{j=1}^m P_n(j) = \sum_{n \in \mathbf{N}} p_n \prod_{j=1}^m \int_0^\infty P_n(j|r) f_{d_j}(r) dr \quad (12)$$

4 Simulation Results

In this section, we present the numerical results for the successful transmission probability of the cache-enabled vehicle network. The accuracy of the analytical results in Sect. 3 is verified by comparing them with the empirical results obtained from Monte-Carlo simulations. We will also discuss the effect of various parameters such as different cache parameters, and density of nodes on the performance of the content caching. The results can provide the design insights for the content caching in vehicular networks.

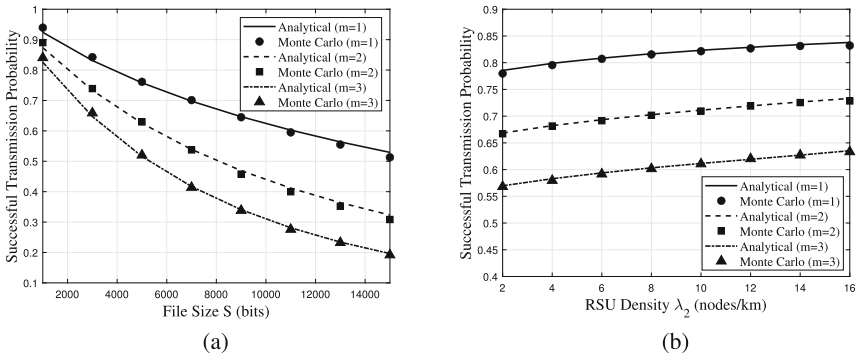


Fig. 2. The impact of the file size S and the RSU density λ_2 ($S = 4000bits$)

In the simulation, we assume that $N = 100$, $\alpha = 4$, $W = 10$ MHz, $T = 1$ ms and $\gamma = 1$. As shown in Fig. 2 (a), we plot the successful transmission probability as a function of file size S . We obtain the successful transmission probability under three different caching parameters with $m = 1, 2, 3$. In these setting, the required storage in RSU equals to NS/m bits. The analytical results match well with the numerical results. It can be clearly seen from the figure that there is a trade-off between the successful transmission probability and the storage requirement for the content caching in RSU. When the available storage increases, the successful transmission probability of the content is higher. In Fig. 2 (b), we observe that the successful transmission probability monotonically increases as the RSU density λ_2 increases. Besides, the analytical results is also verified by the numerical results.

As shown in Fig. 3, we plot the successful transmission probability as a function of content size S under limited cache resources. We relax the constraint that all contents are decomposed into m subfiles and allow different number of subfiles m_i can be created for the content n in the partition procedure. For different combinations of m_i , the storage

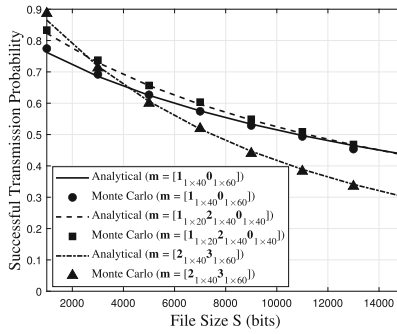


Fig. 3. The impact of the different caching strategies on the content caching.

volume of the RSU is a constant equals to $40 \times S$. We employ three combinations as illustrated in Fig. 3. It can be seen from the figure that when the content size is small, the partition-based cache designs have a higher successful transmission probability. Among the different behaviors, it is interested to find that there exists an optimal combination of m_i to maximize the successful transmission probability. This is the research direction that we are going to explore in the future.

5 Conclusion

In this paper, we consider a cache-enabled vehicular network model with partition-based caching. RLNC scheme is applied for the content coding and SIC is adopted to decode multiple encoded subfiles to recover the request content. We derive the expression for the successful transmission probability with the specific caching parameters by stochastic geometry tools. The result reveal that there is a trade-off between the successful transmission probability and the required storage for content caching in the RSUs. Furthermore, the evaluation revealed the design insights regarding the selection of the RSU density and the caching parameters.

References

1. Fatahi-Bafqi, S., Zeinalpour-Yazdi, Z., Asad, A.: Analytical framework for Mmwave-enabled V2X caching. *IEEE Trans. Veh. Technol.* **70**(1), 585–599 (2021)
2. Moon, H., Baek, K., Ko, I. -Y.: Cache-sharing distributed service registry for highly dynamic V2X environments. In: 2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC), pp. 1123–1124 (2020)
3. Ye, Y., Huang, S., Xiao, M., Ma, Z., Skoglund, M.: Cache-enabled millimeter wave cellular networks with clusters. *IEEE Trans. Commun.* **68**(12), 7732–7745 (2020)
4. Fan, C., Zhang, T., Liu, Y., Zeng, Z.: Cache-enabled HetNets with limited backhaul: a stochastic geometry model. *IEEE Trans. Commun.* **68**(11), 7007–7022 (2020)
5. X. Li, X. Wang and V. C. M. Leung: weighted network traffic offloading in cache-enabled heterogeneous networks. In: 2016 IEEE International Conference on Communications (ICC), pp. 1–6 (2016)

6. Jiang, D., Cui, Y.: Partition-based caching in large-scale SIC-enabled wireless networks. *IEEE Trans. Wirel. Commun.* **17**(3), 1660–1675 (2018)
7. Jiang, Y., et al.: Analysis and optimization of fog radio access networks with hybrid caching: delay and energy efficiency. *IEEE Trans. Wirel. Commun.* **20**(1), 69–82 (2021)
8. Jiang, Y., et al.: Analysis and optimization of cache-enabled fog radio access networks: successful transmission probability, fractional offloaded traffic and delay. *IEEE Trans. Veh. Technol.* **69**(5), 5219–5231 (2020)
9. Dhillon, H.S., Chetlur, V.V.: Foundations and Applications to Vehicular Networks. *Synth. Lect. Morgan Claypool Poisson Line Cox Process* **1**(1), 1–149 (2020)

PHY/MAC/Routing Protocols



A Service Customized Reliable Routing Mechanism Based on SRv6

Peichen Li¹, Deyong Zhang², Xingwei Wang¹ (✉), Bo Yi¹, and Min Huang³

¹ College of Computer Science and Engineering, Northeastern University, Shenyang, China
wangxw@mail.neu.edu.cn

² Software College, Northeastern University, Shenyang, China

³ College of Information Science and Engineering, Northeastern University, Shenyang, China

Abstract. Reliable routing is a classic problem in the field of computer networks. After a network fault occurs, how to choose the recovery path directly determines the performance of network services. This paper introduces service customized techniques into reliable routing. By meeting customized traffic protection requirements, network service quality can be ensured after fault recovery. Topology Independent Loop-free Alternate (TI-LFA) supported by SRv6 is a new reliable routing technology. In this paper, an SRv6-based service customized reliable routing mechanism is designed for the single link failure in the case of P-Q space adjacency in TI-LFA. For traffic with QoS requirements, this paper uses fuzzy theory to make the optimal decision for SRv6 candidate protection schemes. Finally, three representative topologies are selected to build an experimental network supporting SRv6 based on ONOS, Mininet, and the programmable data plane. The results show that when responding to a network service customized request, the recovery path selected by the mechanism proposed in this paper is superior to the comparison mechanism of related QoS indicators.

Keywords: SRv6 · Service customized networks · Reliable routing · Fuzzy theory · TI-LFA

1 Introduction

With the rapid development of the Internet in recent decades, the types of network services are also increasing. As different types of network services have different QoS requirements, Internet Service Providers (ISPs) face new challenges, that is, acquiring the ability to customize networks while providing differentiated network resources. The requirement of service customized networks brings new challenges to network programmability.

SDN is a new network architecture proposed by Stanford University. SDN separates the data plane from the control plane and endows the network with programmability [1]. SRv6 is a specific implementation of segment routing in the IPv6 forwarding plane. SRv6 has all the strengths of segment routing while taking full advantage of the programmable capability and extensibility of IPv6 extension headers [2]. The programmable

data plane increases the network's programmable capability of the data plane, allowing programmable switches to perform a series of processes on packets.

The reliability of IP networks is a classic research problem in the field of computer networks. The main research content is to ensure uninterrupted data transmission. The existing fault protection methods include fast rerouting [3] and network backup [4]. Fast rerouting refers to calculating the recovery path before a fault occurs. When a fault occurs, the data flow will be directly sent to the recovery path. Fast reroute technologies currently include Loop-free Alternate (LFA), Remote Loop-free Alternate (RLFA), and Topology Independent Loop-free Alternate (TI-LFA) algorithms. The TI-LFA mechanism supported by SRv6 is based on the segment routing model. Compared with LFA and RLFA, TI-LFA uses the source routing feature of segment routing to achieve topologically independent loop-free backup. However, TI-LFA only randomly selects the Q node closest to the protected node, which cannot meet the QoS requirements of data flow.

This paper uses the programmable features of SRv6 and TI-LFA to build a service customized reliable routing mechanism so that the network can customize the recovery path for data flow. The scenario of this paper is the single link failure in SRv6 when the P and Q spaces of TI-LFA are adjacent. The aim is to combine the programmable capabilities of SRv6 with SDN and the programmable data plane so that the TI-LFA based on SRv6 can have the capability of service customization. Finally, network services can customize the protection scheme of reliable routing according to their own QoS requirements.

The main contributions of this paper are:

- Based on customized reliable routing requests and standardized network services, our arbitration mechanism in TI-LFA can make the optimal decision for multiple SRv6 protection schemes.
- We propose an SRv6 customized reliable routing algorithm. To meet the QoS requirements of network services, the Fuzzy Analytic Hierarchy Process (FAHP) and Fuzzy comprehensive evaluation are used to select the optimal protection scheme for network services with QoS requirements.
- The cost factor is designed for measuring users' willingness to pay. Our algorithm can select the optimal QoS recovery path for network service within the cost that users can afford.

2 Related Work

SRv6 is a network architecture and P4 is a domain-specific language for programmable data planes. Given the new possibilities brought by SRv6 and P4 language, paper [5] considers the combination of SRv6 and P4 and realizes the forwarding pipeline. The pipeline supports the latest extended uSID instruction of SRv6 by using P4. It also supports the new SRv6 behavior by using the BMv2 software switch and the extended ONOS model. For scheduling experiment-sensitive services, paper [6] presents a method to calculate the segment list, which can guarantee the delay. The algorithm is divided into two steps. Firstly, some nodes in the network are extracted to construct an auxiliary graph

to reduce the complexity of topology. Then, based on the auxiliary graph and weights of links obtained in the first step, the segment list is calculated using the Bellman-Ford algorithm.

When a failure occurs in a network, TI-LFA quickly restores packet forwarding without waiting for other nodes to update their routing tables. However, determining the segment routing sections has a high computational cost because it requires computation for each destination. The paper [7] proposes a method that only computes three times of shortest path tree for TI-LFA rerouting path. This algorithm does not need to calculate each destination separately, which reduces the calculation cost of segmented routing. The paper [8] studies how to use segmented routing technology to enable ISPs to provide connection service reliably through disjoint paths. This paper introduces a robust disjoint path, which can automatically calculate the robust disjoint path based on segmented routing, and proves that the algorithm can be extended to large ISP networks.

In terms of service customized networks and network service composition, relevant scholars design some service models based on specific network service requirements and consider future network architecture. The paper [9] proposes a QoS-aware network cloud service composition method and describes the service composition problem as a variant of the multi-constrained optimal path problem. Finally, an approximate algorithm is used to solve the problem. The paper [10] investigates the new network architecture proposed recently and the emerging technologies to meet the new network needs. Combined with service customized networks, it discusses the opportunities and challenges of the future Internet and guides future network customization.

3 SRv6 Customized Reliable Routing Algorithm Based on Fuzzy Theory

The SRv6 customized reliable routing algorithm uses FAHP, triangular fuzzy number membership function, fuzzy comprehensive evaluation, and TOPSIS method to construct a reliable routing decision model. The cost factor is also defined, which is used to process the customized request with QoS requirements. The customized algorithm replaces the traditional TI-LFA algorithm of randomly selecting the nearest PQ nodes, and adds an arbitration mechanism to it so that the network has the ability of customized reliable routing.

3.1 Reliable Routing Customized Request

According to the ITU standards, this paper defines the key QoS attributes of various network services. Table 1 gives the network service types combined with user overhead. *BW*, *DL*, *JT*, and *LOSS* indicate the network bandwidth, delay, jitter, and packet loss rate respectively. *ID* is used to identify different network services. The primary attribute and the secondary attribute represent the primary and secondary factors affecting users' experience respectively. *Cost* indicates the cost level that users are willing to pay for corresponding network services. The cost of customized services is higher than that of common services.

Table 1. Service type

Service Name	ID	Primary Attribute	Secondary Attribute	Cost
Voice Session	1	<i>DL</i>	<i>JT</i>	1
Voice Session (customized)	2	<i>DL</i>	<i>JT</i>	2
Video Phone	3	<i>JT</i>	<i>LOSS</i>	1
Video Phone (customized)	4	<i>JT</i>	<i>LOSS</i>	2
Streaming Video	5	<i>BW</i>	<i>NULL</i>	1
Streaming Video (customized)	6	<i>BW</i>	<i>NULL</i>	2
Live Video	7	<i>JT</i>	<i>LOSS</i>	1
Live Video (customized)	8	<i>JT</i>	<i>LOSS</i>	2
High Quality Audio	9	<i>JT</i>	<i>LOSS</i>	1
High Quality Audio (customized)	10	<i>JT</i>	<i>LOSS</i>	2

3.2 Fuzzy Analytic Hierarchy Process

FAHP is used to determine the weight of network services. The basis of constructing a fuzzy theory is to determine the form of fuzzy numbers, which can be divided into triangle fuzzy numbers, trapezoidal fuzzy numbers, and interval numbers. The triangular fuzzy number is expressed as (a, b, c) , where a represents the lower limit of a fuzzy number, c represents the upper limit of a fuzzy number, and b represents the most likely value of a fuzzy number. The importance of QoS indicators of network services can be described as “very important”, “important”, “normal”, “unimportant”, and “very unimportant”. The importance of QoS indicators can be described by a triangular fuzzy number, and the mapping relationship is shown in Table 2.

Table 2. Mapping table of triangular fuzzy number and evaluation language set

Evaluation language variable	Triangular fuzzy number
Very unimportant	$(0, 0, 0.25)$
Unimportant	$(0, 0.25, 0.5)$
Normal	$(0.25, 0.5, 0.75)$
Important	$(0.5, 0.75, 1)$
Very unimportant	$(0.75, 1, 1)$

The evaluation matrix EM of FAHP is shown in Formula 1. $e_{x,y}$ represents the importance of the indicator x to the indicator y in the service, and its value is the corresponding triangular fuzzy number. $e_{x,y}$ and $e_{y,x}$ are opposite fuzzy numbers to each other.

$$EM = \begin{pmatrix} (1, 1, 1) & e_{1,2} & \dots & e_{1,m-1} & e_{1,m} \\ \vdots & & \ddots & & \vdots \\ e_{m,1} & e_{m,2} & \dots & e_{m,m-1} & (1, 1, 1) \end{pmatrix} \tag{1}$$

3.3 Calculate QoS Weight

According to users' customized requests, the algorithm will provide a corresponding SRv6 reliable routing protection scheme according to their QoS requirements. The QoS weight of each network service needs to be determined. First, extract each dimension of matrix EM into separate matrices (A, B, C) as shown in Formula 2:

$$(A, B, C) = \left\{ \begin{pmatrix} a_{1,1} & \dots & a_{1,m} \\ \vdots & \ddots & \vdots \\ a_{m,1} & \dots & a_{m,m} \end{pmatrix}, \begin{pmatrix} b_{1,1} & \dots & b_{1,m} \\ \vdots & \ddots & \vdots \\ b_{m,1} & \dots & b_{m,m} \end{pmatrix}, \begin{pmatrix} c_{1,1} & \dots & c_{1,m} \\ \vdots & \ddots & \vdots \\ c_{m,1} & \dots & c_{m,m} \end{pmatrix} \right\} \tag{2}$$

Then compute the *LESS*, *MIDDLE*, and *UPPER* arrays as shown in Formula 3: $(LESS, MIDDLE, UPPER) =$

$$\left\{ \left(\frac{\sum_{i=1}^m a_{1,i}}{\sum_{x=1}^m \sum_{y=1}^m c_{x,y}} \dots \frac{\sum_{i=1}^m a_{m,i}}{\sum_{x=1}^m \sum_{y=1}^m c_{x,y}} \right), \left(\frac{\sum_{i=1}^m b_{1,i}}{\sum_{x=1}^m \sum_{y=1}^m b_{x,y}} \dots \frac{\sum_{i=1}^m b_{m,i}}{\sum_{x=1}^m \sum_{y=1}^m b_{x,y}} \right), \left(\frac{\sum_{i=1}^m c_{1,i}}{\sum_{x=1}^m \sum_{y=1}^m a_{x,y}} \dots \frac{\sum_{i=1}^m c_{m,i}}{\sum_{x=1}^m \sum_{y=1}^m a_{x,y}} \right) \right\} \tag{3}$$

Then takes a value from *LESS*, *MIDDLE*, and *UPPER*, then sorts them as a row in array M , as shown in Formula 4. In each row, *less* is the minimum number, *middle* is the middle number, and *upper* is the maximum number.

$$M = \begin{pmatrix} less_1 & middle_1 & upper_1 \\ \vdots & \vdots & \vdots \\ less_m & middle_m & upper_m \end{pmatrix} \tag{4}$$

Then select any line $M_x = (less_x, middle_x, upper_x)$ in M and every other line $M_y = (less_y, middle_y, upper_y)$ to calculate C_{Number} , as shown in Formula 5:

$$C_{Number} = \begin{cases} 1, & middle_x \geq middle_y \\ \frac{upper_x - less_y}{(upper_x - middle_x) + (middle_y - less_y)}, & upper_x \geq less_y \\ 0, & \text{Others} \end{cases} \tag{5}$$

Finally, a matrix C_{Result} of m rows and $m - 1$ columns is obtained, as shown in Formula 6:

$$C_{Result} = \begin{pmatrix} r_{1,1} & \cdots & r_{1,m-1} \\ \vdots & & \vdots \\ r_{m,1} & \cdots & r_{m,m-1} \end{pmatrix} \quad (6)$$

The minimum value of each line in C_{Result} is selected to generate vector C_{Min} , as shown in Formula 7:

$$C_{Min} = (min_1, min_2, \cdots, min_m) \quad (7)$$

Finally, the weights of indicators are normalized to obtain the *MetricWeight* vector of each indicator, as shown in Formula 8:

$$MetricWeight = \left(\frac{min_1}{\sum_{i=1}^m min_i}, \frac{min_2}{\sum_{i=1}^m min_i}, \cdots, \frac{min_m}{\sum_{i=1}^m min_i} \right) \quad (8)$$

3.4 Cost Factor Design

The cost factor represents the cost that users are willing to pay to customize reliable routing services. This paper mainly considers QoS value, PQ nodes computing resources, and request time.

The more the primary and secondary attributes of the network service meet the requirements of the service, the higher the service price will be. In this paper, normalized values of primary and secondary attributes are used to participate in the calculation of overhead factors, denoted as $Cost_{QoS_1}$, $Cost_{QoS_2}$.

PQ nodes are important computing resources for establishing protection paths in TI-LFA. How PQ nodes are used directly determines the performance of a network. PQ nodes should not be used as an unlimited network resource. The total cost of PQ nodes is denoted in Formula 9. $Cost_P$ and $Cost_Q$ represent the cost of PQ nodes on the protection scheme. $AllCost_{PQ}$ represents the total cost of all PQ nodes in the network.

$$Cost_{PQ} = \frac{Cost_P + Cost_Q}{AllCost_{PQ}} \quad (9)$$

The supply and demand of services is also a concern, that is, “less is more”. There are obvious differences in the popularity of network usage in different periods. The supply capacity of the network changes over time. When Internet resources are scarce, users often need to pay higher prices. In order to quantitatively depict the popularity of network usage in different periods, the percentage data of network users from Baidu [11] were collected. The fitting is divided into five periods, and the fitting function is

shown in Formula 10.

$$F(x) = \begin{cases} 0.034x^3 - 0.433x^2 + 0.84x + 5.065, & 23 : 00 < x \leq 6 : 00 \\ -0.056x^3 + 0.517x^2 - 0.503x + 1.47, & 6 : 00 < x \leq 11 : 00 \\ 0.028x^3 - 0.128x^2 - 0.023x + 5.202, & 11 : 00 < x \leq 13 : 00 \\ 0.007x^3 - 0.169x^2 + 0.873x + 4.047, & 13 : 00 < x \leq 19 : 00 \\ -0.047x^3 + 0.373x^2 - 0.429x + 4.258, & 19 : 00 < x \leq 23 : 00 \end{cases} \quad (10)$$

After obtaining the fitting function, the deviation standardization method is adopted for normalization, and the formula is shown in Formula 11. Where, $F(x)_{min}$ represents the minimum value of $F(x)$, and $F(x)_{max}$ represents the maximum value of $F(x)$.

$$time = \frac{F(x) - F(x)_{min}}{F(x)_{max} - F(x)_{min}} \quad (11)$$

Formula 12 shows the calculation process of the cost factor, which determines the price to be paid for the scheme.

$$Cost = (Cost_{QoS_1} + Cost_{QoS_2} + Cost_{PQ}) * time \quad (12)$$

Based on QoS parameters and cost factors corresponding to the SRv6 protection scheme, protection scheme evaluation matrix DM can be generated, as shown in Formula 13.

$$DM = \begin{pmatrix} bw_1 & dl_1 & jt_1 & loss_1 & Cost_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ bw_n & dl_n & jt_n & loss_n & Cost_n \end{pmatrix} \quad (13)$$

3.5 Customized Decision Based on TOPSIS Method

This paper uses the TOPSIS method to rank the schemes in the judgment matrix DM . The TOPSIS method first selects the global best and worst values for each attribute to form the best and worst points. Then calculate the distance between the points corresponding to each scheme and select the optimal scheme.

Weighted Euclidean distance is used to evaluate each scheme, as shown in Formulas 14 and 15. qos_i represents the bandwidth, delay, jitter, and packet loss rate of the i th scheme in DM . $best_i$ represents the best value of the four indicators, $worst_i$ represents the worst value of the four indicators.

$$D_{best} = \sqrt{\sum_{i=1}^4 w_i (qos_i - best_i)^2} \quad (14)$$

$$D_{worst} = \sqrt{\sum_{i=1}^4 w_i (qos_i - worst_i)^2} \quad (15)$$

Formula 16 is used to calculate the $Topsis_C$ value for each scenario. The scheme with the maximum $Topsis_C$ value is optimal, providing the optimal list of protected segments to meet user customized requests.

$$Topsis_C = \frac{D_{worst}}{D_{worst} + D_{best}} \tag{16}$$

4 Evaluation

4.1 Setup

As shown in Fig. 1, This paper uses the following three representative network topologies for experiments. Topology 1 has a total of 8 nodes and 11 links. Topology 1 selects the

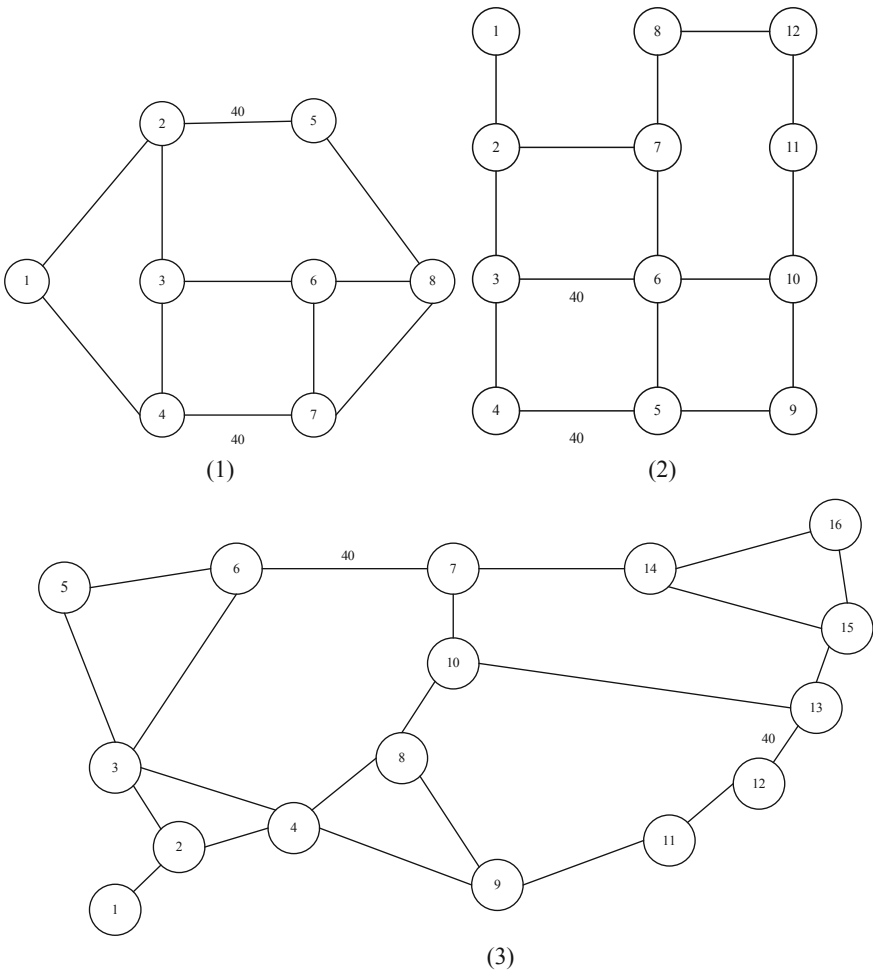


Fig. 1. Topo diagram

topology used in paper [12], which studies network fault recovery and puts forward S3P algorithm. Topology 2 has 12 nodes and 15 links. Topology 2 selects the topology used in the TI-LFA section of *Segment Routing*, which is written by Cisco fellow Clarence Pilsfil, the inventor and promoter of segmented routing. Topology 3 has 16 nodes and 22 links. Topology 3 selects the topology of USA backbone network ANS [13].

4.2 Performance Comparison

In topology 1–3, 30 experiments are performed to calculate the average QoS indicators of the protected path to ensure data stability and reliability.

Bandwidth. In the performance evaluation of bandwidth, streaming video and customized streaming video are selected as network services with high bandwidth requirement, and high quality streaming audio is selected as service with low bandwidth requirement. The experimental results are shown in Fig. 2. The customized protected path bandwidth of streaming video is 9.22% higher than that of high-quality streaming audio, 13.88% higher than that of TI-LFA, and 32.94% higher than that of ordinary streaming video on average.

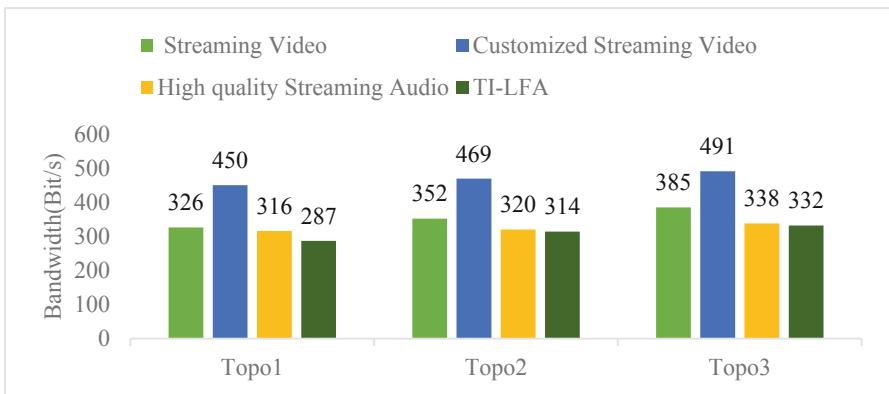


Fig. 2. Bandwidth of different topologies and routing algorithms

Latency. In the performance evaluation of delay, voice session and customized voice session are selected as network services with low delay requirement, and high quality streaming audio is selected as service with high delay tolerance. The experimental results are shown in Fig. 3. The latency of customized protected paths for voice sessions is 4.51% lower than that of high quality streaming audio, 8.13% lower than that of TI-LFA, and 25.9% lower than that of common voice sessions on average.

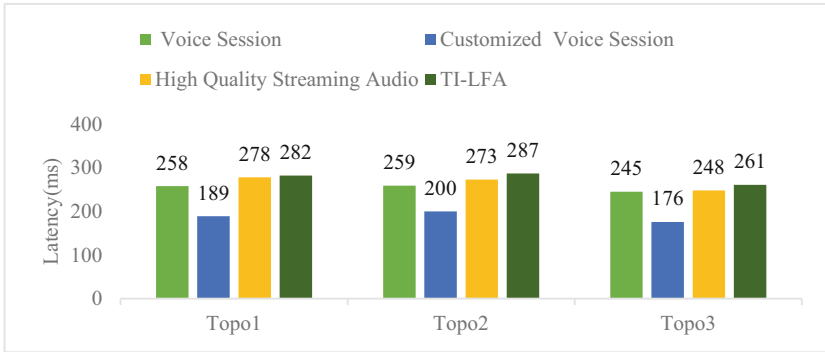


Fig. 3. Latency of different topologies and routing algorithms

Jitter. In the performance evaluation of jitter, high quality streaming audio and customized high quality streaming audio are selected as network services with low jitter requirement, streaming video is selected as service with high jitter tolerance. The experimental results are shown in Fig. 4. The jitter value of customized protection path for high quality streaming audio is calculated to be 12.71%, lower than that for streaming video, 14.32% lower than that for TI-LFA, and 33.68% lower than that for ordinary high-quality streaming audio.

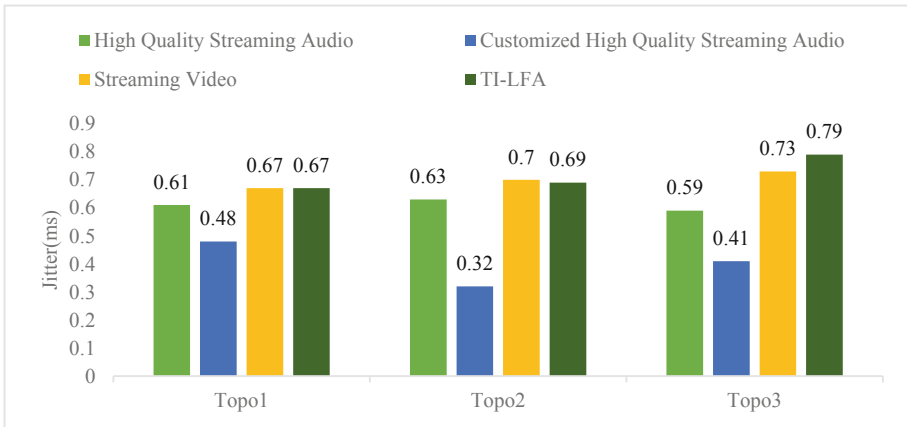


Fig. 4. Jitter of different topologies and routing algorithms

Packet Loss. In the performance evaluation of packet loss rate, video phone and customized video phone are selected as network services with low packet loss rate requirement, and streaming video is selected as services with high packet loss rate tolerance. The experimental results are shown in Fig. 5. The packet loss rate of customized protection path of video phone is 4.4% lower than that of streaming video, 10.11% lower than that of TI-LFA, and 28.96% lower than that of ordinary video phone.

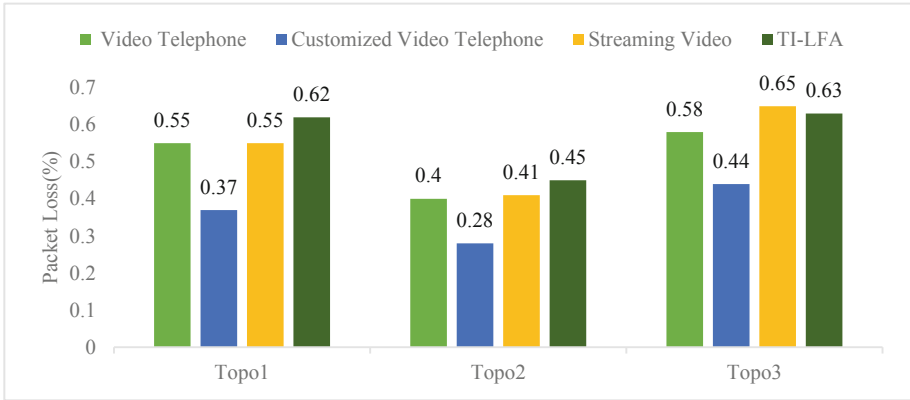


Fig. 5. Packet loss rate of different topologies and routing algorithms

Cost. In the performance evaluation of cost, live video and customized live video are selected for comparison. The experimental results are shown in Fig. 6. The cost of customized protected path of customized video broadcast is 55.77% higher than that of ordinary live video on average and 18.55% higher than that of TI-LFA on average. The algorithm proposed in this paper allows users to obtain better QoS performance by paying more than that of ordinary service.

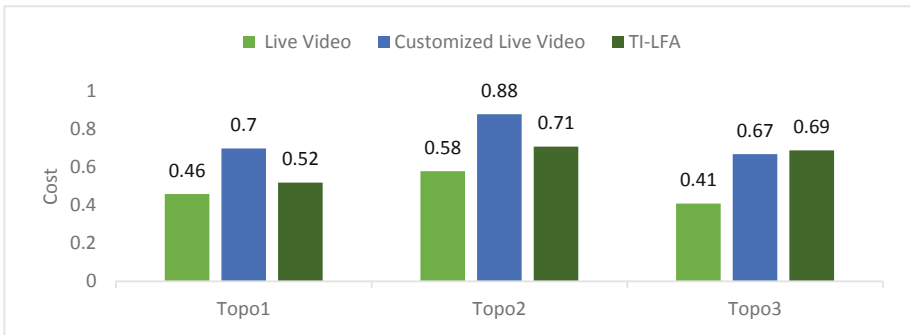


Fig. 6. Cost of different topologies and routing algorithms

5 Conclusion

In this paper, an SRv6-based service customized reliable routing mechanism is proposed. Considering the characteristics of SRv6, a series of mechanisms for network reliability and service customization are designed. The function of service customization for reliable routing is considered emphatically, and the selection of reliable routes is linked with decision making. The results show that the proposed mechanism can improve service customization capability while ensuring network reliability.

Acknowledgement. This work was supported by the National Key Research and Development Program of China under Grant No. 2019YFB1802800; the Liaoning Revitalization Talents Program under Grant No. XLYC1902010; the National Natural Science Foundation of China under Grant No. 61872073, Grant No. 62002055, and Grant No. 62032013.

References

1. Khorsandroo, S., Sanchez, A.G., Tosun, A.S., Arco, J.M., Doriguzzi-Corin, R.: Hybrid SDN evolution: a comprehensive survey of the state-of-the-art. *Comput. Netw.* **192**, 107981 (2021)
2. Filsfils, C., Dukes, D., Previdi, S., Leddy, J., Matsushima, S., Voyer, D.: IPv6 segment routing header (SRH). IETF InternetDraf, <https://tools.ietf.org/html/draft-ietf-6man-segment-routing-header-18> (2019)
3. Papán, J., Segeč, P., Moravčík, M., Kontšek, M., Mikuš, L., Uramová, J.: Overview of IP fast reroute solutions. In: 2018 16th International Conference on Emerging eLearning Technologies and Applications (ICETA). pp. 417–424. IEEE (2018)
4. Hirano, Y., He, F., Sato, T., Oki, E.: Backup network design against multiple link failures to avoid link capacity overestimation. *IEEE Trans. Netw. Serv. Manage.* **17**, 1254–1267 (2019)
5. Abdelsalam, A., Tulumello, A., Bonola, M., Salsano, S., Filsfils, C.: Pushing Network Programmability to the limits with SRv6 uSIDs and P4. In: Proceedings of the 3rd P4 Workshop in Europe, pp. 62–64 (2020)
6. Lai, L., Cui, M., Yang, F.: A segment list selection algorithm based on delay in segment routing. In: 2021 7th International Conference on Computer and Communications (ICCC), pp. 81–87. IEEE (2021)
7. Aubry, F., Vissicchio, S., Bonaventure, O., Deville, Y.: Robustly disjoint paths with segment routing. In: Proceedings of the 14th international conference on emerging networking experiments and technologies, pp. 204–216 (2018)
8. Suzuki, K.: An Efficient Calculation for TI-LFA Rerouting Path. *IEICE Transactions on Communications* (2021)
9. Huang, J., Duan, Q., Guo, S., Yan, Y., Yu, S.: Converged network-cloud service composition with end-to-end performance guarantee. *IEEE Trans. Cloud Comput.* **6**, 545–557 (2015)
10. Zhang, J., Huang, T., Wang, S., Liu, Y.-J.: Future Internet: trends and challenges. *Frontiers of Inf. Technol. Electronic Eng.* **20**(9), 1185–1194 (2019)
11. APP Traffic Platform <https://tongji.baidu.com/research/app> Accessed 24 Mar 2022
12. Zhu, Z., Li, Q., Xia, S., Xu, M.: Caffè: Congestion-aware fast failure recovery in software defined networks. In: 2018 27th International Conference on Computer Communication and Networks (ICCCN), pp. 1–9. IEEE (2018)
13. The Internet Topology Zoo <http://www.topology-zoo.org/dataset.html> Accessed 24 Mar 2022



PAR: A Power-Aware Routing Algorithm for UAV Networks

Wenbin Zhai¹, Liang Liu^{1(✉)}, Jianfei Peng¹, Youwei Ding², and Wanying Lu¹

¹ Nanjing University of Aeronautics and Astronautics, Nanjing, China

{wenbinzhai, liangliu, pengjf, wangyinglu}@nuaa.edu.cn

² Nanjing University of Chinese Medicine, Nanjing, China

ywding@njucm.edu.cn

Abstract. Unmanned Aerial Vehicles (UAVs) have been widely used in both military and civilian scenarios since they are low in cost and flexible in use. They can adapt to a wide variety of dangerous scenarios and complete many tasks the Manned Aerial Vehicles (MAVs) can not undertake. In order to establish connectivity and collect data in large areas, numerous UAVs often cooperate with each other and set up a UAV wireless network. Many multi-hop routing protocols have been proposed to efficiently deliver messages with high delivery ratio and low energy consumption. However, most of them do not consider that the power level of UAVs is adjustable. In this paper, we propose a Power-Aware Routing (PAR) algorithm for UAV networks. PAR utilizes the pre-planned trajectory information of UAVs to compute the encounters at different power levels, and then constructs a power-aware encounter tree to calculate the transmission path with minimum energy consumption from the source to the destination within the delay constraint. Through extensive simulations, we demonstrate that compared with three classic algorithms, PAR significantly reduces the energy consumption and improves the network performance on the basis of ensuring timely delivery of packets.

Keywords: UAV networks · Routing protocol · Energy optimization · Power-aware · Trajectory-based

1 Introduction

In recent years, with the development of sensors, navigation systems and wireless communication technologies, Unmanned Aerial Vehicle (UAV) networks achieve significant performance improvements and have been widely used in both military and civilian scenarios, such as battlefield surveillance, disaster response, farmland monitoring, etc. Due to the agility, versatility, ease of installation and simplicity of operation, UAVs can adapt to a wide variety of dangerous scenarios and complete many tasks that Manned Aerial Vehicles (MAVs) can not undertake.

To establish connectivity in large areas, a great number of UAVs cooperate with each other in the form of clusters and establish a multi-hop UAV network.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022

L. Wang et al. (Eds.): WASA 2022, LNCS 13473, pp. 333–344, 2022.

https://doi.org/10.1007/978-3-031-19211-1_28

Compared with single-hop UAV networks, multi-hop UAV networks can cover a wider range and undertake more complex tasks. Meanwhile, multi-hop UAV networks have many other advantages, such as low cost, low mission completion time, better survivability and better scalability. Many multi-hop routing protocols in UAV networks [10] are proposed to efficiently deliver messages to the destination, which have become a research hotspot in recent years.

Due to the unique characteristics of UAV networks, such as high mobility, sparse distribution, intermittent connectivity and unstable link quality, multi-hop routing in UAV networks faces many challenges, such as low delivery ratio, high delay and expensive energy consumption. In order to address these issues, a mechanism called store-carry-forward (SCF) [1] has been proposed to improve the delivery ratio of packets. With the SCF mechanism, if there is a suitable forwarding node within the communication range, the current message-holder UAV will forward the packet to this forwarding node, otherwise it will store and carry the packet until it encounters a suitable forwarding UAV.

Many routing protocols [1, 8] have been proposed on the basis of the SCF mechanism. However, most of them focus on improving the delivery ratio, but ignoring the delivery delay and energy consumption. In many time-sensitive or delay-constrained applications for UAV networks, such as forest surveillance, disaster rescue and battlefield networks, messages need to be delivered in time, otherwise, their values will be greatly reduced or even invalid. Meanwhile, UAVs are energy-constrained and required to work for a long time. Therefore, routing protocols for UAV networks need to reduce the energy consumption on the basis of ensuring the timely delivery of messages.

Energy-efficient routing has attracted considerable attention and many methods have been proposed to minimize the total energy consumption. However, most of them [4] assume that the power level of UAVs is fixed. In fact, the transmission power of many UAVs is now adjustable [2, 12], and the performance of existing routing protocols is inefficient because they do not utilize the power-adjustable characteristic of UAVs to optimize routing.

Meanwhile, in many military and civilian application scenarios for UAV networks, the trajectories of UAVs are pre-planned and can be obtained in advance through mission planning and path planning [3, 7]. Furthermore, if trajectories of UAVs need to be dynamically adjusted, the ground station can broadcast the updated global trajectory information through the out-of-band channel to ensure that all UAVs have the latest global trajectory information [9, 12]. We can use this pre-planned trajectory information to reduce the energy consumption and ensure the delivery delay.

Therefore, in this paper, we design a Power-Aware Routing (PAR) algorithm for UAV networks which takes the delivery ratio, energy consumption and delay constraint into consideration. The main idea of PAR is to utilize the power-adjustable characteristic and pre-planned trajectory information of UAVs to optimize routing protocols. First, PAR utilizes the pre-planned trajectory information to compute encounters between UAVs at different power levels. Then, a power-aware encounter tree (PET) is constructed according to the encounter

information, based on which PAR can calculate an efficient transmission path with minimum energy consumption within the delay constraint. It is worth noting that, in PAR, each UAV selects the appropriate power level for each packet transmission dynamically and individually. The main contributions of this paper are twofold:

- A power-aware routing algorithm for UAV networks called PAR is proposed to find the efficient transmission path with minimum energy consumption within the delay constraint. Different from existing routing protocols using a fix-power model, for each packet transmission, PAR dynamically selects an appropriate power level for each UAV to reduce the energy consumption on the basis of ensuring timely delivery of the packet.
- Extensive simulations are conducted by using the Opportunistic Network Environment (ONE) simulator [6]. The results show the superior performance of PAR compared to three classic algorithms in terms of the delivery ratio, energy consumption and overhead ratio.

The remainder of the paper is organized as follows. Section 2 summarizes state-of-art on routing protocols in mobile ad-hoc networks (MANETs). In Sect. 3, we describe the problem formulation. Then, in Sect. 4, the design of PAR is provided in detail. Finally, we provide the performance evaluation of PAR through extensive simulations in Sect. 5, and conclude this paper in Sect. 6.

2 Related Work

In this section, we introduce the latest progress of related work. There is a considerable research effort for the development of routing protocols in MANETs, which can be divided into topology-based or geographic routing protocols.

Topology-based routing protocols rely on the current network topology, based on which routing-related information can be obtained and utilized for message forwarding. The authors in [13] propose a new neighbor discovery mechanism and a social network-based relay selection scheme to help make routing decisions. The authors in [5] combine the blockchain with traditional Optimized Link State Routing Protocol to encourage cooperation between nodes. Moreover, a relay selection game model is improved and further applied to select the appropriate relay node. However, the high mobility of nodes and high dynamic of topology in UAV networks make the current topology information out of date frequently and quickly, thereby making the routing inefficient or even unavailable.

Geographic routing protocols exploit local location information instead of global topology information to route data. The pure idea is to forward packets to the neighbor node nearest to the destination. The authors in [4] adaptively utilize the location information and the characteristics of energy consumption to make routing decisions for better route recovery from routing holes. The authors in [8] further consider the transmission direction of data packets while using geographic location information to improve the routing efficiency in similar strip networks. However, due to the sparse distribution of nodes and intermittent connectivity

of communications, pure geographic routing protocols are not enough to cope with UAV networks.

In order to address the above challenges, a promising approach called store-carry-forward mechanism is proposed and widely used due to its simplicity and effectiveness. In [2], the minimum energy routing problem is converted into a directed Steiner tree problem and then map the computed tree back into a transmission scheme. Meanwhile, the authors in [1] further take the prediction of trajectory and the load of UAVs into consideration to optimize data packet forwarding. However, due to the highly dynamic nature of UAV networks, geographic routing protocols inevitably falls into local optimum.

3 Problem Formulation

The application scenarios we consider in this paper are search and rescue missions. Many UAVs search in an area and will send messages as needed. Without loss of generality, we abstract the three-dimensional space into a Euclidean space ignoring the vertical space [2]. The ground station calculates the trajectories of UAVs in advance according to the path planning algorithm [7,9,10].

The UAV network model can be denoted as a weighted directed graph $G = (V, E, P)$ where $V = \{u_1, u_2, \dots, u_N\}$ stands for the N nodes, each node represents a UAV or a ground station. Each node has L adjustable power levels, $P = \{p_1, p_2, \dots, p_k, \dots, p_L\}$, and $E = \{e_1, e_2, \dots, e_G\} \subseteq V \times V$ denotes the edge set. Time is divided into discrete T time slots and nodes remain static within a time slot [2]. An edge $e = (u_i, u_j, t, p_k)$, where $1 \leq i, j \leq N, 0 \leq t \leq T, 1 \leq k \leq L$ and $i \neq j$, means that u_i will encounter u_j and u_i can communicate with u_j with the power level p_k in the time slot t . Moreover, the energy consumption that u_i transmits a packet m to u_j with the power level p_k is denoted as $E_e(p_k)$.

Given a UAV network, every time a real-time message m is generated, it will be associated with a delay constraint T based on its urgency. Taking the power-adjustable characteristic of UAVs into consideration, the objective of this paper is to find an efficient transmission path with minimum energy consumption while satisfying the delay constraint for each message.

4 Power-Aware Routing Algorithm

4.1 Basic Idea

Our basic idea is to utilize the power-adjustable characteristic and pre-planned trajectory information of UAVs to optimize routing in UAV networks. First, the pre-planned trajectory information is used to predict encounters at different power levels between UAVs. As depicted in Fig. 1, there are five UAVs u_1, u_2, u_3, u_4, u_5 and a ground station g_0 . UAVs are flying along with their pre-planned trajectories and collecting data. We assume that each UAV has two different power levels (i.e., p_1 and p_2), and Fig. 1 shows the encounters between UAVs at different power levels (i.e., e_i and c_i). For ease of differentiation, we

abstract encounters at the power level p_1 as points [9]. For instance, at position e_1 , UAV u_1 and u_2 encounter at 10s with the power level p_1 . Moreover, at position c_1 , UAV u_2 and u_3 encounter at 25s with the power level p_2 ; at position c_2 , UAV u_2 and u_5 encounter at 25s with the power level p_2 ; at position c_3 , UAV u_5 and u_2 encounter at 45s with the power level p_2 .

Then, for the message that need to be delivered, we construct a power-aware encounter tree (see in Sect. 4.2) according to the encounter information of UAVs. Based on the power-aware encounter tree, we can find the efficient transmission path with minimum energy consumption within the delay constraint for the message. In PAR, each UAV can select the appropriate power level for each packet transmission dynamically and individually according to its respective encounter situation.

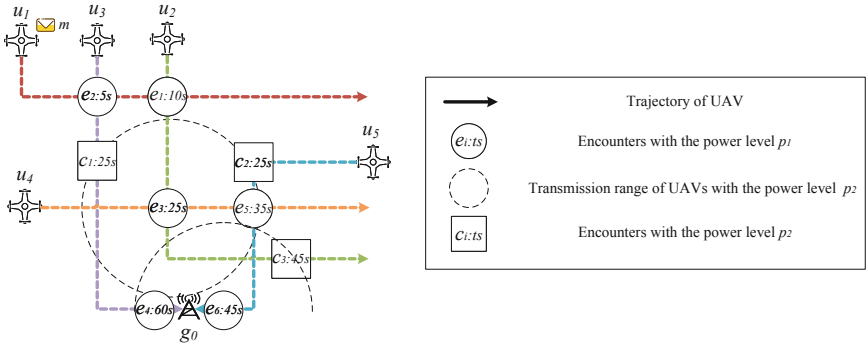


Fig. 1. An example of encounters between UAVs when UAVs have two power levels.

For example, in Fig. 1, a data packet m with a delay constraint T is generated by UAV u_1 at 0s, and u_1 wants to send m to the ground station g_0 . For convenience, in this paper, we use a sublinear energy model in [11], that is $2E_e(p_1) > E_e(p_2)$. It is worth nothing that PAR is not restricted by a specific energy model. According to the pre-planned trajectory information and encounter information, we can deduce that there are at least three transmission paths:

1. $pa_1 : u_1 \xrightarrow{e_2} u_3 \xrightarrow{e_4} g_0$. The transmission path is $(u_1, u_3, 5s, p_1)$, $(u_3, g_0, 60s, p_1)$. The delivery time of m along this path is 60s and the energy consumption is $2E_e(p_1)$.
2. $pa_2 : u_1 \xrightarrow{e_1} u_2 \xrightarrow{e_3} u_4 \xrightarrow{e_5} u_5 \xrightarrow{e_6} g_0$. The transmission path is $(u_1, u_2, 10s, p_1)$, $(u_2, u_4, 25s, p_1)$, $(u_4, u_5, 35s, p_1)$, $(u_5, g_0, 45s, p_1)$. The delivery time of m along this path is 45s and the energy consumption is $4E_e(p_1)$.
3. $pa_3 : u_1 \xrightarrow{e_1} u_2 \xrightarrow{c_2} u_5 \xrightarrow{e_6} g_0$. The transmission path is $(u_1, u_2, 10s, p_1)$, $(u_2, u_5, 25s, p_2)$, $(u_5, g_0, 45s, p_1)$. The delivery time of m along this path is 45s and the energy consumption is $2E_e(p_1) + E_e(p_2)$.

When the delay constraint $T \geq 60\text{s}$, all three transmission paths mentioned above can deliver the message m in time, but the first transmission path pa_1 is the best choice, since it has the least energy consumption while ensuring the timely delivery of m . However, when $45\text{s} \leq T < 60\text{s}$, pa_1 can not satisfy the requirement because the delivery time along it is 60s which exceeds the delay constraint. In this situation, if we do not consider the power-adjustable characteristic of UAVs, the suitable transmission path will be pa_2 , the delivery time of m along pa_2 is 45s and the energy consumption is $4E_e(p_1)$. On the contrary, if we take the adjustable power levels into consideration, we can find out a better transmission path, that is, pa_3 . The delivery time of m along this path is also 45s. Both pa_2 and pa_3 can ensure the timely delivery of m , however, the energy consumption of pa_3 is $2E_e(p_1) + E_e(p_2)$, which is less than that of pa_2 which does not consider the adjustable power level.

4.2 Power-aware Encounter Tree Construction

In order to find an efficient transmission path with minimum energy consumption within the delay constraint for the packet m , we propose a *power-aware encounter tree (PET)* based on the encounter information of UAVs at different power levels within the delay constraint. Before describing the construction procedure of *PET*, we first introduce some basic definitions and terms of *PET*:

- *PET* is a directed tree that originates from the source node s that sends m to the destination node d .
- For a node n in *PET*, it represents a UAV or a ground station in the network. Each node can appear *repeatedly* in *PET*.
- Each edge e between nodes in *PET* is a directed edge, which represents an encounter between the two parties. The direction of the edge indicates the direction of the packet transmission. *Each encounter can only be added to PET once.*
- The child nodes of n are the nodes which n will encounter at different power levels *after encountering its parent node* within the delay constraint. For the same node that encounters at different power levels, we regard it as different child nodes. The child nodes of n are denoted as $n.C = \bigcup_{k=1}^L N(n, n.t, T, p_k)$, where $n.t$ indicates the encounter time between n and its parent node, T represents the delay constraint of m , and $N(n, n.t, T, p_k)$ represents the nodes that n will encounter between $n.t$ and T with the power level p_k .
- For each child node of n , denoted as $c \in n.C$, and its encounter e_c with n , represented as $(n, c, c.t, p_k)$, the transmission path from the source node s to c consists of the transmission path from s to n and the transmission path from n to c , denoted as $c.pa = n.pa \rightarrow (n, c, c.t, p_k)$.
- The total energy consumption to transmit m from s to c along $c.pa$ is denoted as $E_t(c.pa)$, and $E_t(c.pa) = E_t(n.pa) + E_e(p_k)$, where $E_e(p_k)$ indicates the energy consumption that the node n transmits m to its child node c with the power level p_k .
- *PET* only contains the source node s when it is initialized. For the source node s , $s.t = 0$, $E_t(s.pa) = 0$.

Then, we describe the construction procedure of a *power-aware encounter tree*. The construction of *PET* is a process of expanding the tree by adding *new pairs of nodes and edges* into it one by one until a transmission path that satisfies the requirements is found. Each pair of added node n and edge e_n is associated with an energy consumption metric (*ECM*) and encounter time (*ET*), where $ECM = E_t(n.pa)$ and $ET = n.t$. We use a priority queue (*PQ*) to assist the insertion of nodes and edges in *PET*. When each node is added into *PQ*, its parent node and the encounter (i.e., the associated edge) have been determined. Nodes with their edges are sorted by the associated *ECM* and *ET* to ensure that the head node in *PQ* has the smallest *ECM*. Meanwhile, in order to ensure the uniqueness of the encounters in *PET*, we use a bitmap V to mark whether an encounter has been added to *PET*. The algorithm is expressed as follows:

1. Insert the message source UAV into *PQ*; all the elements of V are initialized with 0, indicating that all encounters have not been added into *PET* yet.
2. If *PQ* is empty, the construction process of *PET* is done; otherwise, take out the first node (denoted as u) from *PQ*. If u is the source UAV, go to Step 4; otherwise, get the encounter e_u between u and its parent node, and then go to Step 3. Note that u is the node with the smallest *ECM*.
3. If e_u has been added into *PET* before (i.e., $V[e_u] = 1$), discard it and go to Step 2; otherwise go to Step 4.
4. Based on the pre-planned trajectory information, calculate the child nodes of u (i.e., $u.C$). Note that the child nodes of u are the UAVs that u will encounter at different power levels (i.e., from p_1 to p_L) within the delay constraint T after encountering its parent node (i.e., between $u.t$ and T).
5. For each child node c of u (i.e., $c \in u.C$), compute its *ECM* and *ET* based on the encounter e_c between u and c . If e_c has been added to *PET*, then discard it; otherwise, node c along with e_c is added to *PQ*. The priority queue *PQ* can ensure that the *ECM* of the head node is the smallest.
6. If u is the source node, it will be the root node of *PET*; otherwise, add u into *PET* by inserting it into its parent's corresponding child-list as a child node. The edge represents the encounter between u and its parent node (i.e., e_u). Then, set the corresponding bitmap element $V[e_u]$ to 1, indicating that encounter e_u has been added into *PET*.
7. If the destination node d is added into *PET*, the construction process of *PET* will be done, which means a qualified transmission path is found; otherwise, go to Step 2.

After describing the main stages of the construction of the *power-aware encounter tree*, we summarize PAR in Algorithm 1.

4.3 Proof of Optimality

In this section, we provide a theoretical proof of the optimality of PAR.

Theorem 1. *When the destination node d is added into *PET* for the first time, its associated transmission path (i.e., $d.pa$) will be the transmission path with minimum energy consumption while satisfying the delay constraint.*

Algorithm 1 PAR: A Power-Aware Routing Algorithm for UAV Networks

Require:Source node (s), destination node (d), delay constraint (T)**Ensure:**Power-Aware Encounter Tree (PET)

```

1:  $PQ.push(s)$ , initialize( $V, 0$ )
2: while  $!PQ.isEmpty()$  do
3:    $u \leftarrow PQ.poll()$ 
4:   if  $V[e_u] == 1$  then
5:     continue
6:   end if
7:    $u.C \leftarrow \bigcup_{k=1}^L N(u, u.t, T, p_k)$ 
8:   for each  $c \in u.C$  do
9:     computeECM( $c$ )
10:    computeET( $c$ )
11:    if  $V[e_c] == 0$  then
12:       $PQ.push(c)$ 
13:    end if
14:  end for
15:  insertPET( $u, u.parent, e_u$ )
16:  set  $V[e_u] = 1$ 
17:  if  $u == d$  then
18:    break
19:  end if
20: end while
21: return PET

```

Proof. We suppose that there is another transmission path $d.pa_1$, which consumes less energy than $d.pa$ (i.e., $E_t(d.pa_1) < E_t(d.pa)$) while satisfying the delay constraint. There will be two cases:

1. $d.pa_1$ has been added to PET .

It contradicts the condition that the destination node d is added into PET for the first time.

2. Part or all of $d.pa_1$ is still in PQ .

We assume that the part of $d.pa_1$ in PQ is $u.pa'_1$, so $E_t(u.pa'_1) < E_t(d.pa_1) < E_t(d.pa)$. However, based on the rule of PAR, the ECM of the head node of PQ is the smallest. When d is taken out from PQ and added into PET , the ECM of its associated transmission path (i.e., $d.pa$) is the smallest, namely $E_t(d.pa) < E_t(u.pa'_1) < E_t(d.pa_1)$. Contradiction.

Therefore, there is no other transmission path with lower energy consumption, namely $d.pa$ is the transmission path with minimum energy consumption while satisfying the delay constraint.

Based on the above analysis, we can prove that if there is a transmission path that satisfies the requirements, PAR can always find it. This ensures the correctness and optimality of PAR.

5 Performance Evaluation

In this section, we evaluate the performance of PAR along with three classic routing algorithms in [1]: DTN_{geo} , DTN_{close} and DTN_{load} . DTN_{geo} utilizes the current location and trajectory information for packet forwarding. Then on the basis of the DTN_{geo} algorithm, DTN_{close} further predicts the future location of UAVs, and DTN_{load} considers the load of UAV networks to optimize routing. We implement them on the Opportunistic Network Environment (ONE) simulator [6] and extend the ONE simulator to support multiple power levels.

5.1 Simulation Setup and Scenarios

Referring to the simulation scenarios in [1, 9, 10], we design a power-adjustable simulation scenario inspired by search and rescue missions. In the scenario, one stationary ground station is placed together with nine search UAVs and four ferry UAVs. Each search UAV uses a typical search zigzag movement pattern to cover the mission region efficiently, and each ferry UAV moves back and forth along specified trajectory to assist search UAVs in transmitting packets. Table 1 summarizes the detailed experimental parameters. We use the following metrics to evaluate the performance of the packet forwarding algorithms:

- **Delivery ratio.** The fraction of messages that have been successfully delivered to the destination out of the messages that have been generated.
- **Energy consumption.** The average energy consumed by messages successfully delivered to their destinations.
- **Overhead ratio.** $Overhead = \frac{Sum_{relay} - Sum_{delly}}{Sum_{delly}}$, where Sum_{relay} is the total times that all messages were forwarded, and Sum_{delly} is the total number of messages that have been successfully delivered to the destination.

Table 1. Simulation settings

Parameter	Default value
Simulation area (m^2)	800 × 800
Simulation time (s)	480
Number of nodes	14
UAV speed (m/s)	4.5
Number of power levels	4
Communication range (m)	200
Message size (<i>Byte</i>)	1400
Message creation Rate of Each UAV ($/s$)	6
Delay constraint (s)	75

To avoid bias, we run each experiment with 10 rounds and calculate the average value as the final experimental result. In addition, for DTN_{geo} , DTN_{close} and DTN_{load} , we run the simulation separately at each fixed power level and select the best simulation value as the final experimental result.

5.2 Analysis of Simulation Results

Impact of the Message Creation Rate. As shown in Fig. 2(a), PAR achieves the maximal delivery ratio and outperforms the three algorithms especially when the message creation rate becomes very high. This is because PAR utilizes the pre-planned trajectory information and power-adjustable characteristic of UAVs to calculate the transmission path in advance which improves the delivery ratio of data packets. Meanwhile, as depicted in Fig. 2(b), the energy consumption of PAR is much lower than DTN_{geo} and DTN_{load} , but higher than DTN_{close} . However, DTN_{close} is at the expense of the delivery ratio, while PAR firstly guarantees the timely delivery of messages, and then minimizes the energy consumption. PAR can improve the power level and increase the energy consumption to ensure the timely delivery of messages. For the overhead ratio, as Fig. 2(c) shows, PAR achieves the minimum overhead ratio and there is almost no fluctuation in PAR. This is because PAR calculates the transmission paths in advance, and then forwards the messages according to these calculated paths without redundant forwarding, thereby reducing the number of message forwarding.

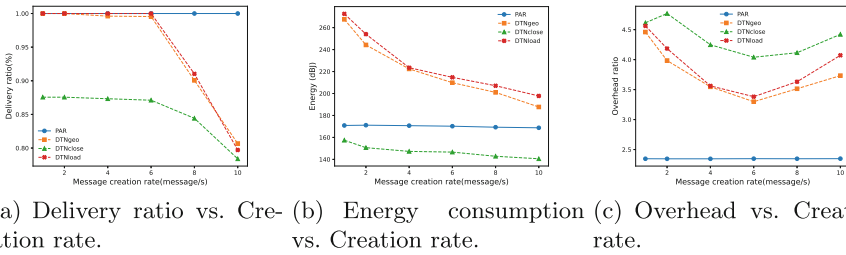
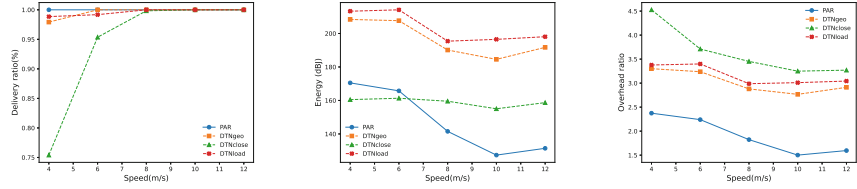


Fig. 2. The impact of the message creation rate on the delivery ratio, energy consumption and overhead ratio.

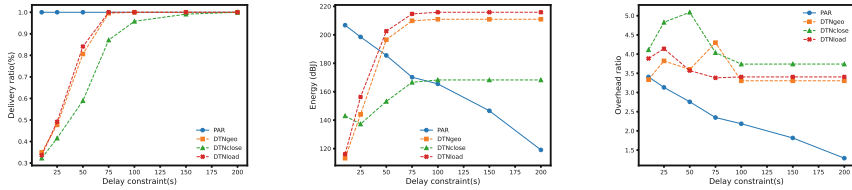
Impact of the UAV Speed. As depicted in Fig. 3(a), the delivery ratios of all algorithms increase with the increasing of the UAV speed. This is because with the increasing of the UAV speed, the current message holder can have more opportunities to choose a suitable forwarding node before the delay constraint expires due to more encounters in the same time window. The energy consumption of PAR is always lower than DTN_{geo} and DTN_{load} , but higher than DTN_{close} when the UAV speed is slower than 8m/s, as Fig. 3(b) shows. This is because PAR dynamically adjust the power of UAVs to find an efficient transmission path to the destination. On the contrary, DTN_{close} does not guarantee the timely delivery of packets. When the delivery ratios of PAR and DTN_{close} are the same, we can find that the energy consumption of PAR is much smaller than DTN_{close} . As shown in Fig. 3(c), the overhead ratios of all algorithms decrease with the increasing of the UAV speed, and PAR achieves the minimum overhead ratio.



(a) Delivery ratio vs. Speed. (b) Energy consumption vs. Speed. (c) Overhead vs. Speed.

Fig. 3. The impact of the UAV speed on the delivery ratio, energy consumption and overhead ratio.

Impact of the Delay Constraint T . As Fig. 3(a) and 3(b) shows, even when the delay constraint is small, PAR can still maintain a perfect delivery ratio. The reason is that PAR adjusts (i.e., increases) the power level to ensure the timely delivery of messages. Meanwhile, as the delay constraint tends to be relaxed, PAR gradually adjusts (i.e., decreases) the power level to minimize the energy consumption. For the overhead ratio, as depicted in Fig. 3(c), PAR achieves the minimum overhead ratio, and as the delay constraint increases, the overhead ratio of PAR gradually decreases. This is because PAR always chooses the transmission path with minimum energy consumption within the delay constraint (Fig. 4).



(a) Delivery ratio vs. Delay constraint. (b) Energy consumption vs. Delay constraint. (c) Overhead vs. Delay constraint.

Fig. 4. The impact of the delivery constraint on the delivery ratio, energy consumption and overhead ratio.

6 Conclusion

In this paper, we propose an efficient Power-Aware Routing (PAR) algorithm for UAV networks. PAR takes the adjustable power into consideration. Based on the pre-planned trajectory information of UAVs, PAR computes encounters between UAVs at different power levels and constructs a power-aware encounter tree to find an efficient transmission path. Then according to the found transmission path, it selects the appropriate power level for each forwarding UAV to minimize the energy consumption and ensure timely delivery of the packet. The simulation

results show that PAR significantly improves the network performance and has a better delivery ratio, energy consumption and overhead ratio than three classic algorithms.

Acknowledgment. This work is supported by the National Natural Science Foundation of China under No. U20B2050, Public Service Platform for Basic Software and Hardware Supply Chain Guarantee under No. TC210804A, the “National Key R&D Program of China” under No. 2021YFB2700500 and 2021YFB2700502.

References

1. Asadpour, M., Hummel, K.A., Giustiniano, D., Draskovic, S.: Route or carry: motion-driven packet forwarding in micro aerial vehicle networks. *IEEE Trans. Mob. Comput.* **16**(3), 843–856 (2016)
2. Fu, L., et al.: Joint optimization of multicast energy in delay-constrained mobile wireless networks. *IEEE/ACM Trans. Networking* **26**(1), 633–646 (2018)
3. Hsu, Y.H., Gau, R.H.: Reinforcement learning-based collision avoidance and optimal trajectory planning in UAV communication networks. *IEEE Trans. Mobile Comput.* **1**(1), 1–149 (2020)
4. Huang, H., Yin, H., Min, G., Zhang, J., Wu, Y., Zhang, X.: Energy-aware dual-path geographic routing to bypass routing holes in wireless sensor networks. *IEEE Trans. Mob. Comput.* **17**(6), 1339–1352 (2017)
5. Kadadha, M., Otok, H.: A blockchain-enabled relay selection for QoS-OLSR in urban VANET: a stackelberg game model. *Ad Hoc Networks* **117**, 102502 (2021)
6. Keränen, A., Ott, J., Kärkkäinen, T.: The one simulator for DTN protocol evaluation. In: *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, pp. 1–10 (2009)
7. Li, X., Liu, L., Wang, L., Xi, J., Peng, J., Meng, J.: Trajectory-aware spatio-temporal range query processing for unmanned aerial vehicle networks. *Comput. Commun.* **178**, 271–285 (2021)
8. Liu, C., Fang, D., Hu, Y., Tang, S., Xu, D., Cui, W., Chen, X., Liu, B., Xu, G., Chen, H.: Easygo: low-cost and robust geographic opportunistic sensing routing in a strip topology wireless sensor network. *Comput. Netw.* **143**, 191–205 (2018)
9. Peng, J., Gao, H., Liu, L., Li, N., Xu, X.: TBM: an efficient trajectory-based multicast routing protocol for sparse UAV networks. In: *2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pp. 867–872. IEEE (2020)
10. Peng, J., Gao, H., Liu, L., Wu, Y., Xu, X.: Fntar: a future network topology-aware routing protocol in UAV networks. In: *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6. IEEE (2020)
11. Zamalloa, M.Z., Seada, K., Krishnamachari, B., Helmy, A.: Efficient geographic routing over lossy links in wireless sensor networks. *ACM Trans. Sens. Netw. (TOSN)* **4**(3), 1–33 (2008)
12. Zhang, G., Wu, Q., Cui, M., Zhang, R.: Securing UAV communications via joint trajectory and power control. *IEEE Trans. Wirel. Commun.* **18**(2), 1376–1389 (2019)
13. Zhang, L., Hu, L., Hu, F., Ye, Z., Li, X., Kumar, S.: Enhanced OLSR routing for airborne networks with multi-beam directional antennas. *Ad Hoc Networks* **102**, 102116 (2020)



Multi-Channel RPL Protocol Based on Cross-Layer Design in High-Density LLN

Jianjun Lei¹(✉), Tianpeng Wang¹, Xunwei Zhao², Chunling Zhang²,
Jie Bai², Zhigang Wang², and Dan Wang²

¹ School of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing 400065, China

leijj@cqupt.edu.cn

² State Grid Information and Telecommunication Group CO., LTD, Beijing 102211, China

{zhaoxunwei,zhangchunling,baijie2,
wangzhigang,wangdan2}@sgitg.sgcc.com.cn

Abstract. Low power and lossy network (LLN) massive terminal deployment has become an inevitable trend. However, traditional routing protocols cannot meet the large-scale data transmission requirements. In this paper, we introduce the multi-channel communication technology into LLN and propose a multi-channel routing protocol based on cross-layer design (MC-RPL), which can increase the data transmission capacity of the network via a parallel data transmission strategy. Specifically, we design a novel super-frame structure to decouple the communication period into a route maintenance phase and a data transmission phase. Nodes can transmit data in parallel during the data transmission phase. Besides, we improve the trickle algorithm to enhance routing maintenance efficiency during the route maintenance phase. Simulation results have demonstrated the effectiveness of the MC-RPL protocol compared to the MRHOF and IRH-OF protocols.

Keywords: LLN · Multi-channel · Routing protocol · Cross-layer · Superframe structure

1 Introduction

Low Power Lossy Network Routing Protocol (RPL) is a distance vector routing protocol proposed by the Low Power Lossy Network Routing Working Group (ROLL), which builds network topology based on Objective Function [1]. Due to the sharp increase in wireless sensor equipment brought about by diversification of practical application scenarios, network density is getting crucially

This work was supported by the key cooperation project of Chongqing municipal education commission(HZ2021008).

high and the limited capacity of LLN networks has become a bottleneck of network performance. Explore multi-channel communication in wireless networks has been long known as a practical way to increase network capacity as well as mitigate interference in radio links. A multitude of nodes can transmit data packets in multiple radio channels simultaneously, thus appealing multi-channel communication mode in LLN is also could consider a promising approach. During recent decades, several multi-channel protocols have been proposed to optimize the performance of the single-channel deployment in legacy WSN. However, the majority of existing multi-channel research in wireless sensor networks is mainly performed on the Media Access Control (MAC) layer while there are seldom studies focusing the routing layer optimization based on multi-channel scenarios. Routing protocols based on multi-channel scenarios have much more outstanding performance compared with single-channel and become a main trend in the future. In this paper, we propose a multi-channel routing protocol, called MC-RPL. The main contributions are as follows:

- We design a new superframe structure to support multi-channel communication in the LLN network, in which system communication time is divided into the initialization phase, data transmission phase, and route maintenance phase.
- In the data transmission phase, we design a channel allocation mechanism based on the pseudo-random code of the MAC address, which can improve the data transmission performance of the network by enabling different channels for parallel data transmission for multiple data link pairs.
- In the route maintenance phase, we optimize the trickle algorithm to adjust the sending period of control messages, which can accelerate the network convergence and avoid unnecessary overhead. In addition, the multi-channel of MAC mechanism enhances the reliability of data transmission, while optimizing the selection of the optimal parent node.

The rest of the paper is organized as follows: Sect. 2 reviews related work, Sect. 3 presents the proposed MC-RPL protocol, Sect. 4 describes the simulation and result evaluation, and Sect. 5 offers concluding remarks.

2 Related Work

The current research on RPL routing protocol mainly focuses on load balancing and optimizing the RPL protocol by using better routing metrics. IETF-ROLL has developed two objective functions: OF0 [2] and MRHOF [3]. However, network performance may reach the bottleneck when the number of nodes increases to a threshold in single-channel communication mode. Therefore, the multi-channel communication mode in LLN has become a main trend of research. To make better use of the multi-channel communication mode in LLN, the RPL routing protocol needs to combine with the MAC layer for further improvement. The work in [4, 5] conducted a comprehensive analysis of cross-layer protocols

in existing wireless sensor networks and proved that RPL could supply interoperability with other layer protocols. Hassani A E et al. [6] designed a new objective function to improve network performance by considering the Radio Strength Signal Indicator and hop count. Nordin N et al. [7] proposed a multi-channel cross-layer routing protocol (MCRP) that allows physically close nodes to communicate on different channels. This is mostly found in routes for nodes that are not interfered with or have low interference. However, in this scheme channel allocation is performed after topological stabilization, which does not have good robustness. Amitangshu P et al. [8] proposed a distributed dynamic channel and route selection method to optimize network performance. However, all nodes need to maintain the information table of surrounding nodes to reselect the channel in the second stage, resulting in excessive overhead in the network. Bizagwira H et al. [9] divided the entire communication time into three phases: synchronization, data transmission, and sleep. However, the time slot-based communication method may reduce network reliability in high-density scenarios.

In response, this paper proposes the multi-channel routing protocol based on cross-layer design, which can replace the traditional RPL protocol in the high-density multi-channel LLN network. This protocol allows data to be transmitted in parallel via multi-channel at specific stages of the superframe and periodic routine maintenance ensures the stability of the network topology.

3 MC-RPL Protocol

3.1 System Model

In this section, we introduce the system model of the MC-RPL protocol. The MC-RPL protocol takes a single Destination-Oriented Directed Acyclic Graph (DODAG) in a single RPL instance as the main research object. There is only one sink node in the network, and other nodes use the sink node as the root node to construct the network topology. Except for the sink node, all nodes in the entire network are operated in non-storage mode. In addition to being responsible for sending the data packets generated by itself, a non-root node also acts as a relay node to forward the data packets received from the child nodes. In addition, each node that joins the network can maintain clock synchronization with the parent node to ensure that all nodes in the network have a common reference time [10].

3.2 MC-RPL Protocol Design

This section proposes a multi-channel RPL protocol based on a cross-layer design. This protocol guarantees route maintenance and data transmission in LLN networks through the superframe mechanism. Then, the superframe structure, initialization phase, data transmission phase, and route maintenance phase are introduced in detail, respectively.

Superframe Structure. We design a novel superframe structure for the LLN network, splitting the communication period into two parts: the initialization phase and the superframe period. When the node starts the initialization phase, the sink node will start the superframe period after completing the initialization phase within a certain period, and other nodes will estimate the start time of the superframe period according to the parent node after completing the networking in the initialization phase.

A superframe consists of a data transmission phase and route maintenance phase, and each superframe starts with the data transmission phase. In the data transmission phase, each node listens to its working channel and utilizes multi-channel for data packet transmission. In the route maintenance phase, all nodes listen to the control channel and can receive broadcast control information such as DIO (DODAGs Information Object) sent by all surrounding nodes to assure the construction and maintenance of network topology. The superframe structure is shown in Fig. 1.

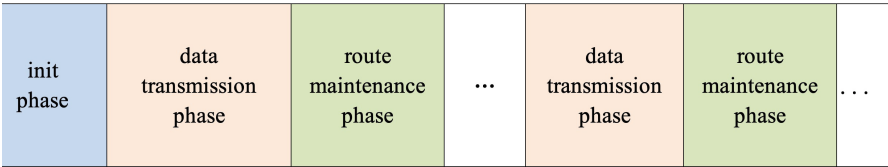


Fig. 1. Superframe structure

Initialization Phase. In the initialization phase, all nodes work on the control channel and initialize their network parameters. After the node completes initialization, the sink node will start to broadcast the DIO control message on the control channel to trigger the node to start network construction. The sink node will start the superframe period after completing the pre-defined initialization phase. In this stage, the superframe index (sf_{index}) of the sink node will be calculated according to Eq. 1 after each node is connected to the network. Each node calculates the start time of the next superframe ($next_sf$) according to Eq. 2 and enters the superframe period with the sink node at the next superframe. The initialization period can ensure that the network topology converges faster when the network starts up.

$$sf_{index} = \left\lfloor \frac{curr_time - init_time}{sf_time} \right\rfloor \tag{1}$$

$$next_sf = sf_{index} \cdot sf_time + init_time \tag{2}$$

Let sf_{index} represent the superframe index of the sink node. $curr_time$ represents the current time of the node, $init_time$ is the duration of the initialization phase and sf_time is the duration of a superframe. $next_sf$ represents the time when the node enters the superframe.

Data Transmission Phase. In the data transmission phase, Multi-channel communication could enable multiple pairs of nodes to communicate in parallel on different channels. Simultaneously, the node will reduce the collision probability of data packets by suppressing the sending of broadcast messages in this phase.

In this phase, each node selects its working channel through a procedure that pseudo-randomly according to its own MAC address. Therefore, when a node communicates with another node, it can estimate the working channel according to the address of the destination node, avoiding the overhead of maintaining node channel information. We have utilized Eq. 3 to calculate the working channel index of a node.

$$Channel_{index}^i = \text{mod}(MacAdd_i, K) + 1 \quad (3)$$

where $Channel_{index}^i$ is the working channel index of node i , $MacAdd_i$ is the MAC address of node i , K is the number of available channels. $\text{mod}(a, b)$ is remainder of a divided by b .

All nodes listen to their working channels, once the source node wants to send a data packet to the destination node, according to Eq. 3, the working channel of the destination node will be calculated. After that, the source node will switch to the corresponding channel for data transmission.

Route Maintenance Phase. In the multi-channel scenario, nodes could transmit data packets on multiple working channels respectively. Since network maintenance requires nodes to broadcast DIO and send other control messages to surrounding neighbors during the route maintenance phase, it is necessary to ensure that all nodes exchange control messages on the same channel, which is called the control channel. The main task of the route maintenance phase is to update and maintain the network topology. In this phase, each node switches the control channel to listen to the control channel sent by the surrounding nodes.

In this phase, each node needs to receive DIO control messages sent by other nodes in the network to join the network, or determine whether to switch from the current parent to a better one or not. In other words, RPL broadcasts the routing information using DIO messages which are transmitted based on the Trickle Algorithm. In MC-RPL protocol, DIO control messages are only sent to the control channel during the route maintenance phase to ensure that all nodes update network information and avoid collision with data transmission. To accelerate the convergence of the network, the node adopts a standard trickle algorithm to construct the network topology during the initialization phase. Once the node enters the superframe period, the optimized Trickle algorithm described above will be activated and inhibit the nodes from sending broadcast data packets during the data transmission phase. For this purpose, the Trickle timer controls the sending of DIO messages and will be frozen during the data transmission phase.

In the standard trickle algorithm, the minimum transmission interval of DIO is $I_{min} = 2^{12}$ ticks, and the maximum transmission interval $I_{max} = 2^{20}$ ticks.

Tick is the relative time unit of the system, also known as the time base of the system, derived from the periodic terminal of the timer. To ensure DIO messages are only transmitted once in a superframe period, the minimum transmission interval of DIO messages needs to be modified properly. The newly defined minimum interval is calculated via Eq. 4:

$$I_{min} = 2^{\lfloor \log_2 R.time \rfloor} \quad (4)$$

where $R.time$ represents the duration of the route maintenance phase. In addition, according to the traditional trickle algorithm, the maximum sending interval of DIO is set to $I_{max} = I_{min} \cdot 2^8$.

In addition, the node analyzes the information carried in the received DIO control message, selects the optimal parent node according to the objective function, and determines whether to switch the parent node. We adopted the objective function named Expected Transmissions Count (ETX) as the routing metric. The MC-RPL protocol can improve the success rate of data packet transmission by introducing multi-channel in the data transmission stage. Therefore, multi-channel communication can ensure that the ETX is relatively stable, thereby improving the stability of the network topology.

4 Experimental Results

To evaluate the performance of the MC-RPL protocol, we implemented the MC-RPL in the Contiki-ng operating system and analyzed its performance with the Cooja simulation platform. In the simulation experiment, the performance of MC-RPL has been investigated through its comparison with MRHOF [3] and IRH-OF [6]. We mainly evaluate from four dimensions: packet delivery rate, end-to-end delay, the throughput of the sink, and control packet overhead. We analyzed the network performance under different node densities, and all nodes were randomly deployed in an area of 200 m \times 200 m. Contiki-ng operates in the 915 MHz band and has a total of 10 available channels, one of which is a control channel. To ensure the accuracy of experimental results, all simulation tests were averaged 5 times, and the simulation time was 30 min. Table 1 describes the summary of the simulation environment.

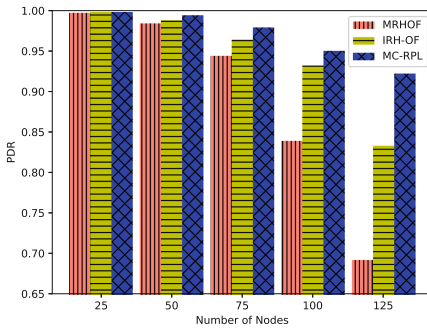
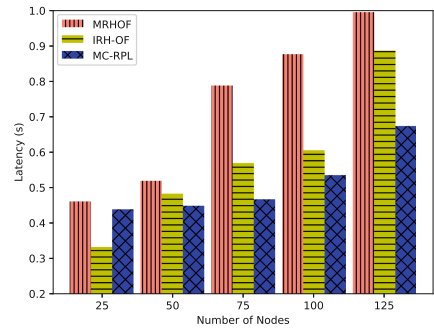
4.1 Packet Delivery Rate

Packet Delivery Rate (PDR) refers to the ratio of the number of data packets successfully received by the destination node compare with the total number of data packets sent. With the network scale getting larger, plenty of packets need to be transmitted in the network, the collision probability of data packets will increase and the PDR declines significantly. In Fig. 2, we compared the PDR of three different protocols. The traffic load is relatively light in the case of low-density topologies for each node. The MRHOF and IRH-OF method can easily cope with this scenario, and the MC-RPL protocol does not have great advantages. However, when the topology gets relatively dense, especially when

Table 1. Simulation parameters

Parameters	Value
Radio environment	Unit disk graph medium (UDGM)
Standard for PHY and MAC	IEEE 802.15.4/CSMA
Number of available channels	10
Duration of initialization phase	20 s
Duration of data transmission phase	9 s
Duration of route maintenance phase	1 s
Network area	200 m \times 200 m
Number of nodes	25,50,75,100,125
Transmission/interference ranges	60/100 m
Data sending rate	6pkt/mins
Package size	127 bytes
Simulation time	30 mins

the number of nodes is 125, the PDR of the MRHOF protocol is reduced to 70% and the IRH-OF protocol is 83%. The reliability of the network is hard to guarantee. In contrast, MC-RPL can still achieve a PDR of more than 92% in this high-density scenario, ensuring network reliability. Results of simulation experiments verify that the MC-RPL protocol in multi-channel communication mode, allowing multiple pairs of nodes to transmit data in parallel, which can effectively alleviate collisions in the network and increase the success rate of data transmission.

**Fig. 2.** Packet delivery rate**Fig. 3.** End-to-end latency

4.2 End-to-End Latency

The end-to-end latency is the duration for the data packet to be transmitted from the source node to the sink node. As shown in Fig. 3, in a network with a relatively limited number of nodes (25 nodes), the MC-RPL protocol needs to perform channel switching during data transmission, resulting in higher delay than IRH-OF protocol and lower delay than MRHOF. With the increase in the number of nodes, MC-RPL can mitigate the average amount of delay against MRHOF and IRH-OF by 31.35% and 15.15%, respectively. The main reason is that only a pair of nodes are allowed to communicate within the node’s listening range in a single channel. If the channel is busy, data packets need to be backed off, which also to increased delay. Nonetheless, MC-RPL allows multiple pairs of nodes to use different channels for simultaneous transmission in multi-channel communication mode.

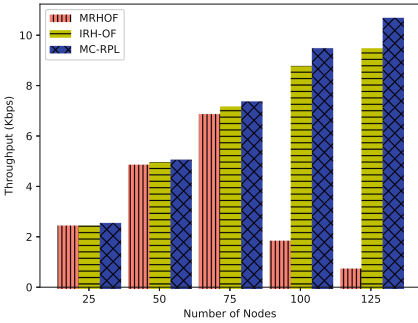


Fig. 4. Throughput

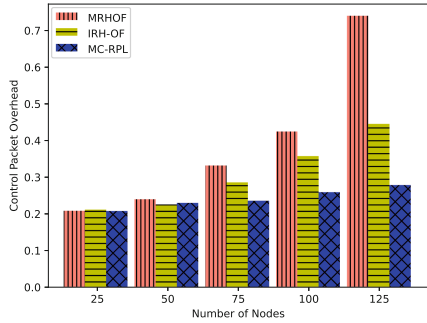


Fig. 5. Control packet overhead

4.3 Throughput

The throughput is the number of data packets successfully transmitted per unit of time. In Fig. 4 can be seen that as the number of nodes increases, the throughput of the IRH-OF protocol and MC-RPL protocol are constantly increasing. The throughput of MC-RPL is always higher than that of IRH-OF. Conversely, the throughput of the MRHOF protocol decreases when the number of nodes exceeds 100. since nodes are always suffering heavy traffic and the data packets cannot be successfully transmitted.

4.4 Control Packet Overhead

Control packet overhead is the ratio of control packets to the total number of generated packets by the nodes. According to Fig. 5, with deploying more nodes in the network, the overhead of MC-RPL is slight to both MRHOF and IRH-OF and has an average reduction of 43% compared to MRHOF. The result

shows that by enabling a multi-channel approach, the MC-RPL protocol, which avoids network topology instability caused by excessive ETX fluctuations, has successfully reduced the retransmission of data packets. Besides, MC-RPL has an average reduction of 28% compared to IRH-OF, due to the adjustment of the trickle algorithm ensures that control messages can be sent in time and minimizes the unnecessary overhead when the network topology is stable.

5 Conclusions

The main challenge faced by the highly densely deployed LLN network is the limited network capacity. To solve this problem, we propose a multi-channel routing protocol called MC-RPL. This protocol supports the construction and maintenance of topology in the multi-channel communication mode by using the superframe structure. The simulation experiment on Cooja shows that the MC-RPL protocol has more satisfactory performance than MRHOF and IRH-OF in terms of PDR, end-to-end delay, throughput, and control overhead.

References

1. Alexander, R., et al.: Rpl: Ipv6 routing protocol for low-power and lossy networks. In: RFC 6550 (2012)
2. Thubert, P.: Objective function zero for the routing protocol for low-power and lossy networks (RPL). RFC 6552 (2012)
3. Gnawali, O., Levis, P.: The minimum rank with hysteresis objective function. RFC 6719 (2012)
4. Kim, H.S., Ko, J., Culler, D.E., Paek, J.: Challenging the ipv6 routing protocol for low-power and lossy networks (RPL): a survey. *IEEE Commun. Surv. Tutorials* **19**(4), 2502–2525 (2017). <https://doi.org/10.1109/COMST.2017.2751617>
5. Fahmy, H.: Cross-layer protocols for wsns. concepts, applications, experimentation and analysis of wireless sensor networks (2020)
6. Hassani, A.E., Sahel, A., Badri, A.: IRH-of: a new objective function for RPL routing protocol in IoT applications. *Wireless Pers. Commun.* **119**(1), 1–17 (2021)
7. Nordin, N., Clegg, R.G., Rio, M.: Multichannel cross-layer routing for sensor networks. In: International Conference on Telecommunications (2016)
8. Amitangshu, P., Asis, N.: Distributed routing and channel selection for multi-channel wireless sensor networks. *J. Sens. Actuator Networks* **6**(3), 10 (2017)
9. Bizagwira, H., Toussaint, J., Misson, M.: Multi-channel routing protocol for dynamic WSN. In: 2016 Wireless Days (WD), pp. 1–3 (2016)
10. Yigitler, H., Badihi, B., Jantti, R.: Overview of time synchronization for IoT deployments: clock discipline algorithms and protocols. *Sensors* **20**(20), 5928 (2020)



Routing Protocol Based on Improved Equal Dimension New Information GM(1,1) Model

Jian Shu^(✉), Hongjian Zhao, and Huanfeng Hu

Nanchang Hangkong University, Nanchang, China
shujian@nchu.edu.cn

Abstract. Aiming at the problems of high end-to-end transmission delay and low packet delivery rate caused by the high mobility of unmanned aerial vehicle (UAV) nodes, a routing protocol based on the improved equal dimension new information GM(1,1) model (IEDNI-GM) is proposed. By analyzing the motion characteristics of the UAV node, combine the gray prediction model and the Markov chain model to construct IEDNI-GM to predict the location of the UAV node at the next moment. Meanwhile, the paper combines the advantage that clustering structure can optimize network management. We consider the motion state and the communication link state between nodes and use the predicted value of node position to calculate the value of link holding time, motion similarity and expected transmission count. The cluster-head election indicator is constructed by combining these three values, and the UAV nodes in the network are clustered. This clustering structure is adopted to improve the AODV routing protocol. Therefore, the source node can find an effective communication route to the destination node. Experiments under the network simulator NS-3 show that compared with routing protocols such as AODV and AODV-ETX, the routing protocol in this paper can effectively reduce the end-to-end average transmission delay, increase the delivery rate of data packets, and is more suitable for UANET.

Keywords: UAV ad hoc networks · Position prediction · Cluster-head election indicator · Clustering routing protocol

1 Introduction

With the continuous expansion of the application field of UAV (Unmanned Aerial Vehicle) [1], the tasks it needs to perform also become more diverse. On some complex occasions, a single UAV system is no longer suitable, and multiple UAVs need to coordinate and cooperate to complete more complex tasks.

Each UAV is equipped with sensors, positioning systems and autopilots. UAVs can relay remote control commands from satellites or ground base stations to each other, and exchange awareness, node energy consumption, and airframe conditions. UANET (UAV Ad Hoc Networks) [2] has the characteristics of high node mobility and frequent network topology changes. This easily leads to problems such as low data delivery rate and high information transmission delay when nodes communicate in the network [3]. The main contributions of this paper are as follows:

- The IEDNI-GM (Improved Equal Dimension New Information GM(1,1) Model) is proposed to predict the position of the UAV node at the next moment. By analyzing the motion characteristics of UAV nodes, IEDNI-GM is constructed by combining GM(1,1) and Markov chain models.
- A location prediction-based UANET clustering routing protocol AODV-PPC is proposed. The predicted node positions are used to calculate the link retention time, motion similarity and expected transmission times between nodes. The cluster head election indicator is constructed by combining the link retention time, motion similarity and expected transmission times between nodes. According to the cluster head election indicator, the nodes in the network are clustered, thereby improving the AODV routing protocol. This improvement can effectively reduce the average end-to-end transmission delay and improve the packet delivery rate.

2 Related Work

This paper focuses on location-based routing protocol [4]. This kind of routing protocol is suitable for UANET, whose topology changes frequently but it is more dependent on the accuracy of the positioning system [2]. Greedy Perimeter Stateless Routing (GPSR) [5] is a typical location-based routing protocol. Each node in the network can get its own location information through the positioning system, and get the location of the target node by constantly interacting with neighbors. This protocol is prone to routing vulnerabilities in networks with rapidly changing topologies and sparse nodes. Rodrigues for the high mobility of UAV nodes. Rodrigues et al. [6] aimed at the high mobility of UAV nodes. The contact duration between nodes is calculated by predicting the location of nodes. The uncertainty of a node is measured by its neighbor's progress in completing its flight plan. Combining the two methods mentioned above to improve GPSR. Sang et al. [7] proposed an opportunistic routing protocol based on trajectory prediction. The protocol predicts the speed and position of the UAV node at the next moment by using a Gaussian mixture model. Calculate the trajectory metric of the node using the predicted value of the node position. This protocol combines node trajectory measurement, node energy and node buffers to make routing decisions. Hussein et al. [8] proposed a geographical multicast routing protocol in the flying ad hoc network. When a node has information to send to multiple destination nodes, it will predict the location of its neighbors, thus obtaining the neighbors and each destination node. If this value is smaller than the transmission radius of the node, it will forward the route through its neighbor. When the node can not find a neighbor closer to the target node than itself, it will forward the boundary until the current node finds a neighbor closer to the target node.

3 Proposed Approaches

This paper designs a UANET routing protocol by predicting the location of UAV nodes. This clustering structure has the advantages of improving the stability of the network and optimizing the management [9]. In this paper, the cluster head election indicator is constructed by combining the link holding time between nodes, the similarity of movement

and expected transmission times. According to the cluster head election indicator, cluster the nodes in the network. So far, the design of the cluster routing protocol AODV-PPC has been completed.

3.1 Node Location Prediction

Equal Dimension New Information GM(1,1) Model (EDNI-GM). EDNI-GM [10] keeps the input dimension constant by employing a sliding time window on the basis of GM(1,1). The time-varying sequence $x^{(0)}(k)$ of original UAV node position data is fed into the EDNI-GM model to obtain the predicted position sequence $X^{(0)}(k + 1)$, as shown in Eq. (1).

$$X^{(0)}(k + 1) = (1 - e^a)(x^{(0)}(1) - \frac{b}{a})e^{-ak}. \tag{1}$$

Among them, a and b are the gray parameters of the model, which can be obtained by the least square method.

Improved EDNI-GM based on GM (1,1). The gray prediction residual value at time k is the difference between the actual position value $x^{(0)}(k)$ of the node at time k and the position prediction value $X^{(0)}(k)$ of EDNI-GM. The predicted value $E^{(0)}(k + 1)$ at the next moment is obtained by inputting the prediction residual data sequence $e^{(0)}(k)$ into GM(1,1), as shown in Eq. (2).

$$E^{(0)}(k + 1) = (1 - e^{a_e})(e^{(0)}(1) - \frac{b_e}{a_e})e^{-a_ek}, \tag{2}$$

Among them, a_e and b_e are the gray parameters of the residual gray prediction model, and the corrected node position prediction value is shown in Eq. (3).

$$X_{GM}^{(0)}(k + 1) = X^{(0)}(k + 1) + param * E^{(0)}(k + 1). \tag{3}$$

where *param* is the correction coefficient.

Improved EDNI-GM based on Markov chain. The Markov chain model can better predict the random fluctuation data series [11]. The Markov chain model divides the residual sequence $E^{(0)}(k + 1)$ and constructs the state transfer matrix according to its characteristics. Thus, the most likely residual state of the predicted data at the latter position is predicted, and the revised results of the Markov chain model are obtained. Taking the node position prediction result curve after GM(1,1) correction as the benchmark, draw a $h + 1$ curve parallel to it. Each two adjacent curves constitute an interval, and each interval constitutes a prediction state $H_r (r = 1..h)$. p_{fg} is the transition probability from state H_f to state H_g , as shown in Eq. (4).

$$p_{fg} = \frac{m_{fg}}{m_f}, \tag{4}$$

Among them, m_{fg} is the number of times the revised residual data is transferred from the state H_f to the state H_g , and m_f is the number of times the state H_f appears in the

revised residual data. The transition probabilities between all states can form a state transition probability matrix.

The corrected node position prediction residual value will be transferred from the state H_f to the state H_l at the next moment. When the initial state vector W_0 is transferred through the l step to obtain the predicted state vector $W_l = W_0 * p$, and there is $\max(p_{fr}) = p_{fl}$ in the f row of the state transition probability matrix p . At this time, the correction value $X_{MG}(k+1)$ of the Markov chain model is the median of the interval corresponding to the state H_l , as shown in Eq. (5)

$$X_{MG}(k+1) = \frac{r_{up} + r_{down}}{2}, \quad (5)$$

Among them, r_{up} is the upper limit of the interval state to which H_l belongs, and r_{down} is the lower limit of the interval state to which H_l belongs.

$X_{GM}(k+1)$ is the gray predicted value corrected by GM(1,1). $X_{MG}(k+1)$ is the corrected value obtained through the Markov chain model. $X_{final}(k+1)$ is the final predicted node position, as in Eq. (6).

$$X_{final}(k+1) = \frac{X_{GM}(k+1)}{1 - X_{MG}(k+1)}. \quad (6)$$

3.2 Constructing Cluster Head Election Indicator

The cluster head election indicator is constructed by link retention time, motion similarity and expected transmission times between nodes. Figure 1 is a schematic diagram of the relative motion between two UAV nodes. There are UAV nodes V_I and V_J in the network. The positions of nodes V_I and V_J are K and A . The velocity vectors of nodes V_I and V_J . Node V_J flies in the direction of AB relative to node V_I . The communication radius of the drone node is R .

Motion Similarity. The motion similarity $Sim(\Delta m_{IJ})$ between the nodes V_I and V_J is the product of the direction similarity $Sim(\Delta \theta_{IJ})$ and the velocity similarity $Sim(\Delta v_{IJ})$, as shown in Eq. (7).

$$Sim(\Delta m_{IJ}) = Sim(\Delta \theta_{IJ}) \times Sim(\Delta v_{IJ}) = \cos \frac{\Delta \theta_{IJ}}{2} \times \frac{1}{1 + \Delta v_{IJ}}, \quad (7)$$

Among them, $\Delta \theta_{IJ}$ is the angle between the motion directions of the nodes V_I and V_J , Δv_{IJ} is the speed difference between the nodes V_I and V_J .

Link Hold Time. The link retention time represents the communication time that the communication link between nodes can provide for two nodes. d_{AB} is the length of the line segment AB , and T_{IJ} is the link retention time between nodes V_I and V_J , as shown in Eq. (8).

$$T_{IJ} = \frac{d_{AB}}{|\vec{v}_J|}, \quad (8)$$

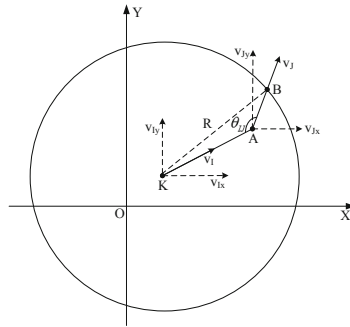


Fig. 1. Schematic diagram of relative motion between two UAV nodes

Expected Transmission Times [5] . The expected number of transmissions ETX_{IJ} between UAV nodes V_I and V_J is shown in Eq. (9).

$$ETX_{IJ} = \frac{1}{s_{IJ-f} \times s_{IJ-r}}, \tag{9}$$

Among them, s_{IJ-f} is the success probability of forward transmission on the link between nodes V_I and V_J , and s_{IJ-r} is the success probability of reverse transmission on the link between nodes V_I and V_J .

Cluster Head Election Indicator. The cluster head election indicator is constructed by combining the link retention time, motion similarity and expected transmission times between nodes and their neighbors in UANET. The cluster head election indicator of the node is shown in Eq. (10).

$$CH_I = \frac{1}{u} \sum_{j=1}^u (T_{Ij} \times (Sim(\Delta m_{Ij}) + ETX_{Ij})). \tag{10}$$

Among them, u is the degree of node V_I , T_{Ij} is the normalized link retention time value between node V_I and its j neighbor, and $Sim(\Delta m_{Ij})$ is the normalized motion similarity value between node V_I and its j neighbor, ETX_{Ij} is the normalized expected number of transmissions between node V_I and its j neighbor.

Cluster Formation. The UAV periodically broadcasts the information packet with its own cluster head election indicator value to the neighbors, and compares it with the neighbor’s cluster head election index value. If its own election value is larger, it continues to broadcast the information packet to the neighbors; If its own election value is small, it stops broadcasting. When a node no longer receives the information packet with the cluster head election index value broadcast by its neighbor, it is elected as the cluster head.

The cluster head node broadcasts its cluster head announcement packet to its neighbors. After receiving the packet, the neighbors send a request packet requesting to join the cluster to the node that issued the cluster head announcement. After receiving the

request packet, the cluster head node replies to the response packet agreeing to join the cluster. The node that sent the request packet becomes a member node of the cluster after receiving the response packet.

After the cluster is formed, the cluster member node with the second largest cluster head election index value becomes the gateway node. The status transition of the UAV node is shown in Fig. 2.

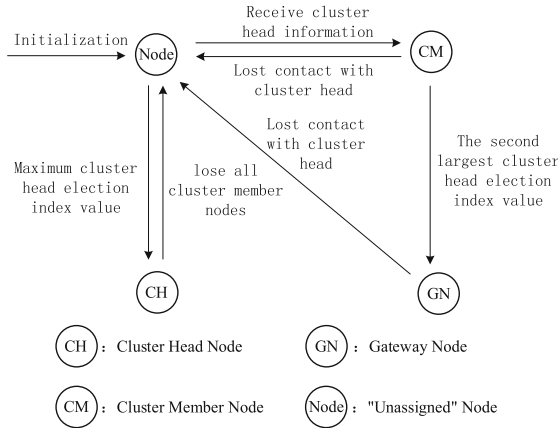


Fig. 2. Schematic diagram of UAV node state transition

4 Simulation Experiment

In the simulation experiment, the performance of the routing protocol is compared and analyzed by using Ubuntu18.04 under the Linux system combined with NS-3. The simulation scene is a rectangular area with a size of 1500m*1500m.

The comparison methods in this paper are AODV [12], AODV-ETX [13], AODV-EE-Hello [14], AODV-PLR-ETX [15], AODV-L-ETX [15]. In the experiment, the performance of routing protocol will be evaluated by packet transmission rate [16], the average end-to-end transmission delay [16] and the throughput [16].

Node Location Prediction Performance Analysis

Figure 3 shows that the prediction errors of IEDNI-GM and EDNI-GM both reach a large value when the movement state of the nodes changes. During the movement process from the 20th moment to the 26th moment, IEDNI-GM and EDNI-GM begin to converge gradually, and the prediction error of IEDNI-GM is smaller. This is because IEDNI-GM is using GM(1,1) to modify EDNI-GM. Markov chain correction is also used to improve the model, so as to strengthen its prediction ability of random item data, which leads to better prediction performance when the node motion state changes. GM(1,1)' s prediction result curve is three time intervals behind the actual node motion state change

due to the continuous influence of previous data, and it is difficult to converge, so its prediction error is too large.

Compared with GM(1,1) and EDNI-GM, IEDNI-GM proposed in this paper has better prediction performance and stronger convergence ability when the movement state of nodes changes, so it is more suitable for predicting the location of nodes in UAV ad hoc networks.

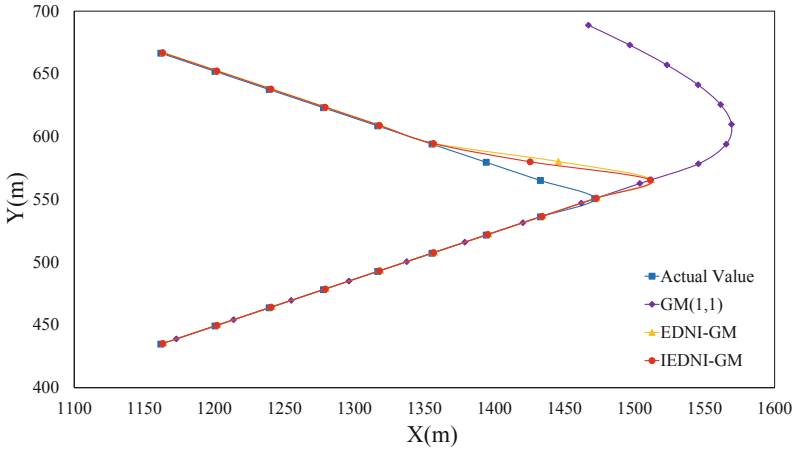


Fig. 3. Comparison of node location prediction

Routing Protocol Performance Analysis

In this experiment, AODV-PPC is compared with other routing protocols by changing the number of UAV nodes in the network.

Figure 4 shows that AODV-PPC performs better in most cases. The reason is that in the AODV-PPC routing protocol, nodes communicate through the cluster head, which avoids the congestion caused by the frequent broadcast of Hello messages, reduces network conflicts, and improves packet delivery rates performance. AODV-ETX, AODV-L-ETX, and AODV-PLR-ETX make routing decisions by measuring the quality of communication links between nodes and are not suitable for networks with frequent topology changes and frequent disconnection of communication links.

Figure 5 shows that AODV-PPC performs the best. The reason is that AODV-PPC builds clusters by considering the communication link state and motion state between nodes, which reduces the impact of unreliable links on data transmission, suppresses network congestion, and reduces the average end-to-end transmission delay. AODV-ETX, AODV-L-ETX and AODV-PLR-ETX do not consider the frequent disconnection of links between nodes in the network, resulting in poor average end-to-end transmission delay performance.

Figure 6 shows that as the number of nodes increases, the network throughput of all routing protocols increases. The throughput performance of the AODV-PPC routing protocol increases the most and remains optimal after the number of nodes reaches

40. The AODV-PPC routing protocol by clustering nodes in the network, even if the number of nodes increases, the routing protocol can reduce the competition and conflict between nodes, thereby improving its throughput performance. Compared with AODV-ETX, AODV-L-ETX, and AODV-PLR-ETX routing protocols, it is easier to establish a route from the source node to the target node in the process of route discovery, to ensure its throughput performance.

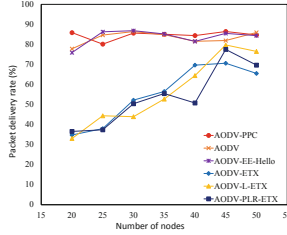


Fig. 4. Packet delivery rate under different number of nodes

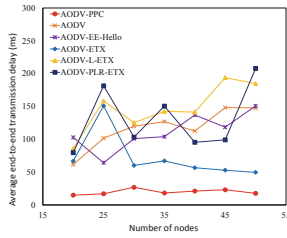


Fig. 5. Average end-to-end transmission delay with different number of nodes

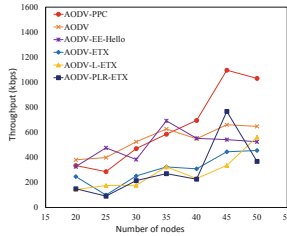


Fig. 6. Throughput with different number of nodes

5 Conclusion

In this paper, IEDNI-GM model is established by analyzing the motion characteristics of UAV nodes. The location of the UAV node at the next moment is predicted through IEDNI-GM. The network nodes are clustered according to the cluster head election

indicator. The cluster head selection indicator is constructed by combining the link holding time between nodes, the similarity of motion and the expected transmissions times. The AODV routing protocol is improved by using the clustering structure, and the AODV-PPC routing protocol based on node location prediction is obtained. With NS-3, the routing protocol is simulated and compared from the two aspects: the moving speed of nodes and the number of nodes. The experimental results show that the AODV-PPC routing protocol has good packet delivery rate, average end-to-end transmission delay and throughput performance. The next step is to improve the selection indicator of cluster head.

Acknowledgment. This paper is supported by the National Natural Science Foundation of China (62062050, 61962037, 61762065), the Innovation Foundation for Postgraduate Student of Jiangxi Province (YC2021130), and the Jiangxi Provincial Natural Science Foundation (20202BABL202039).

References

1. Al-Turjman, F., Abujuubeh, M., Malekloo, A., et al.: UAVs assessment in software-defined IoT networks: an overview. *Comput. Commun.* **150**, 519–536 (2020)
2. Oubbati, O.S., Atiqzaman, M., Lorenz, P., Tareque, M.H., Hossain, M.S.: Routing in Flying Ad Hoc networks: survey, constraints, and future challenge perspectives. *IEEE Access* **7**, pp. 81057–81105 (2019)
3. Arafat, M.Y., Moh, S.: Routing protocols for unmanned aerial vehicle networks: a survey. *IEEE Access* **7**, 99694–99720 (2019)
4. Agrawal, J., Kapoor, M.: A Comparative study on geographic-based routing algorithms for flying ad-hoc networks. *Concurrency and Computation: Practice and Experience* pp. 1–20 (2021). <https://doi.org/10.1002/cpe.6253>
5. Karp, B., Kung, H.T.: GPSR: greedy perimeter stateless routing for wireless networks. In: 2000 Annual International Conference on Mobile Computing and Networking (MobiCom), New York, NY: ACM, pp. 243–254 (2000)
6. Rodrigues, A., Reis, A.B., Sargento, S.: GPSR-PPU: greedy perimeter stateless routing with position prediction and uncertainty for FANETs. In: 2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), Piscataway, NJ: IEEE, pp. 1–6 (2020)
7. Sang, Q., Wu, H., Xing, L., et al.: An energy-efficient opportunistic routing protocol based on trajectory prediction for FANETs. *IEEE Access* **8**, 192009–192020 (2020)
8. Hussen, H.R., Choi, S.C., Park, J.H., et al.: Predictive geographic multicast routing protocol in flying ad hoc networks. *Int. J. Distrib. Sens. Netw.* **15**(7), 1–20 (2019)
9. Arafat, M.Y., Moh, S.: A survey on cluster-based routing protocols for unmanned aerial vehicle networks. *IEEE Access* **7**, 498–516 (2019)
10. Zeng, B.: Equal dimension new information GM(1,1) model and its application. In: 2012 International Conference on Artificial Intelligence and Soft Computing (ICAISC), Berlin, German: Springer, 2012: 535–538
11. Yang, Y., Ke, B.: Research on location prediction of moving objects based on grey markov model. In: Huang, D.-S., Huang, Z.-K., Hussain, A. (eds.) ICIC 2019. LNCS (LNAI), vol. 11645, pp. 213–224. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26766-7_20

12. Perkins, C.E., Belding-Royer, E.M.: ad-hoc on-demand distance vector routing. In: 1999 IEEE Workshop on Mobile Computing Systems and Applications (WMCSA), Los Alamitos, CA: IEEE Computer Society pp. 90-100 (1999)
13. Jevtić, N.J., Malnar, M.Z.: Implementation of ETX metric within the AODV protocol in the NS-3 simulator. *Telfor Journal* **10**(1), 20–25 (2018)
14. Mahmud, I., Cho, Y.Z.: Adaptive hello interval in FANET routing protocols for green UAVs. *IEEE Access* **7**, 63004–63015 (2019)
15. Jevtic, N.J., Malnar, M.Z.: Novel ETX-based metrics for overhead reduction in dynamic ad hoc networks. *IEEE Access* **7**, 116490–116504 (2019)
16. Wang, Q.W., Qi, Q., Cheng, W., Li, D.: Node degree estimation and static game forwarding strategy based routing protocol for ad hoc networks. *J. Software* **31**(6), 1802–1816 (2020)

Algorithms, Systems, and Applications of Edge Computing



An Asynchronous Federated Learning Optimization Scheme Based on Model Partition

Jing Xu^{1,3}, Lei Shi^{1,3(✉)}, Yi Shi², Chen Fang^{1,3}, and Juan Xu^{1,3}

¹ School of Computer Science and Information Engineering,
Hefei University of Technology, Hefei 230009, China
shilei@hfut.edu.cn

² Department of ECE, Virginia Tech, Blacksburg, VA 24061, USA

³ Ministry of Education, Engineering Research Center of Safety Critical Industrial
Measurement and Control Technology, Hefei 230009, China

Abstract. Federated learning based on edge computing environment has great potential to facilitate the implementation of artificial intelligence at the edge of the network. However, because of the limited resource at the edge, place the complete Deep Neural Networks (DNN) model on the edge for training may not a good choice. In this paper, we study the time optimization for asynchronous federated learning based on model partition. That is, the DNN model is divided into two parts and deployed separately on the device and the edge server for the model training. First, we give the metric of the relationship between learning accuracy and iteration frequency, and then we build a mathematical model based on this. Because the solution space of mathematical model is too large to be solved directly, we propose an algorithm to minimize the total time by dynamically adjusting the model partition point and bandwidth allocation. Simulation results show that our algorithm can reduce the time by 32% to 60% compared with the other three methods.

Keywords: Federated learning · Model partition · Edge computing

1 Introduction

Federated learning (FL) is a new machine learning framework [1]. The main idea of FL is to train models on clients, and then send model parameters to the server for parameter aggregation, that it can protect data privacy [2, 3]. Nowadays there are two different federated optimization schemes that have been studied widely: the synchronous FL [4] and the asynchronous FL [5]. In synchronous FL, all participating clients use local data for training the local model, and the server waits for all clients to send the updated local model and aggregates them into

The work is supported by the Major science and technology projects in Anhui Province, No. 202003a05020009.

a new global model. In asynchronous FL, the server does not need to wait for all clients to complete local training, but updates the global model as soon as it receives a local model of any participating client.

In FL framework, learning accuracy is an important performance indicator [6, 7], and it is mainly related to the number of iteration and aggregation strategy [8]. In [9], authors derived the lower bound of local and edge iterations for distributed approximate Newton (DANE) algorithm with given accuracy. In [10], for federated averaging (FedAvg) algorithm running Stochastic Gradient Descent (SGD), authors established a convergence rate of $\mathcal{O}(\frac{1}{T})$ for strongly convex and smooth problems, where T is the iteration frequency of SGDs. In addition, due to the participation of multiple clients, the learning efficiency is an overall performance indicator of FL. In [11, 12], authors studied joint user selection and resource allocation in federated learning in order to improve performance. In [13], authors proposed a device scheduling and resource allocation algorithm based on channel conditions and the importance of local model to improve the FL efficiency and performance. Notice that these methods mentioned above are based on synchronous FL. The efficiency of asynchronous FL can also be studied, and some researchers have done their efforts on this field. In [14], authors proposed an asynchronous federated learning algorithm, which can accelerate convergence. In [15], authors considered the uncertainty and resource limitation of the system, and proposed an asynchronous learning-aware transmission scheduling algorithm for improving learning performance.

Meanwhile, edge computing (EC) is a new computation model [16]. EC pushes computing tasks and services to the edge close to the data source [17]. Therefore, the cooperation between EC and FL can promote the realization of artificial intelligence at the edge of network [18]. By doing this, the benefits of a shared model trained from rich data can be gained, and the computing resources of user devices can be utilized [19]. However, the edge has the disadvantage of limited resources, so the calculation of the complete DNN model at the edge will lead to greater pressure on the edge. Fortunately, the model partition technology can divide the complete DNN model by taking advantage of the structure characteristics, and deploy different parts in different locations, thus reducing the calculation pressure at the edge and reducing the delay [20]. In [21], authors proposed a method to dynamically adjust the partition point for the training stage of DNN at the single server and single device for reducing the end-to-end delay.

Previous works have studied the optimization problem in FL, but there are few researches on asynchronous FL after model partition. In this paper, we consider asynchronous FL based on model partition. Firstly, the model accuracy is converted into the iteration frequency through experiment, and then a mathematical model is established to express the total training time. Since the mathematical model contains a large number of variables and is difficult to be solved directly, we propose a heuristical algorithm to optimize the training time by dynamically adjusting the partition point and bandwidth allocation.

The rest of this paper is organized as follows: In Sect. 2, we introduce our system model and define our problem. In Sect. 3, we analyze the problem and give our algorithm. In Sect. 4, we give the simulation results and analyze them. In Sect. 5, we summarize this paper.

2 System Model and Problem Definition

2.1 System Model

Consider a network with one edge server and N edge devices. Define d_0 as the edge server and $d_i (\in N)$ as one edge device, as shown in Fig. 1. Suppose edge devices are heterogeneous, which means each device has different calculation ability. Assume devices are connected to the server with wireless network. To protect the data privacy, the raw collected training data by devices will not be exchanged among them. Suppose all edge devices start training at the same time, which is 0. We want to train a DNN model by training tasks on the edge server and edge devices, and each device will produce and only produce one training task in the whole training process. Suppose this DNN model has v layers, and denote $l_r (r = 1 \dots v)$ as one of the layer. We use model partition technique for training this model, which means each task will first be trained several layers in its generated edge device, and then be transmitted to the edge server for training the remained layers. Since iterations exist in the training process, so these tasks may be calculated repeatedly in these edge devices and the edge server. When they are in the server, since the server can only process one task at a time, which means when there are multiple tasks to be processed, other tasks cannot be processed until the current task is completed. After E local iterations, model parameters will be updated in the server, and we assume the updating parameter work is also be done on the edge server. We will use the asynchronous updating method, which means when local iterations of a task is completed, model parameters will be transmitted to the server and updated immediately instead of waiting for other iterations to complete. Denote m_i^j as the j -th iteration from d_i , then m_i^j is made up of the following steps:

- 1) d_i performs forward propagation before the partition point and then transmits the intermediate data to the edge server;
- 2) d_0 performs forward propagation and back propagation after the partition point, and transmits the intermediate data of back propagation to d_i ;
- 3) d_i performs backward propagation; if it has reached E local iterations, uploads model parameters on the device to the server for parameter aggregation;
- 4) d_i downloads the new parameter and completes the parameter update.

We want to design the partition method for each iteration and the bandwidth allocation for uploading data to the server to minimize the total training time on the premise of ensuring the learning accuracy.

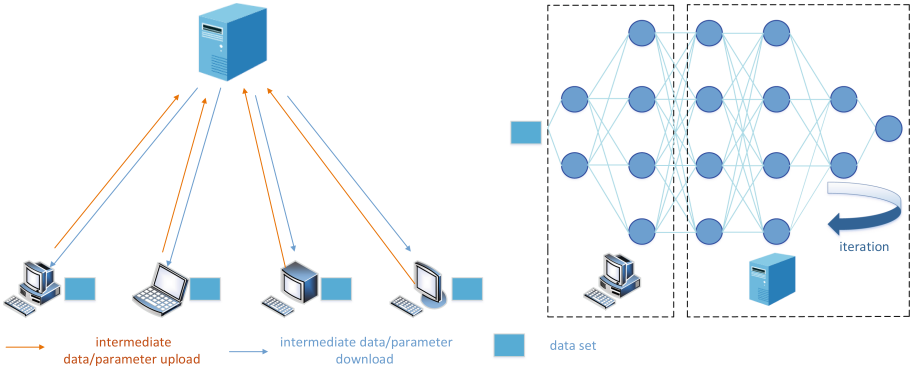


Fig. 1. The system model

2.2 Problem Formulation

We want to minimize the total time T_{total} for the whole training process, which is decided by the task with the maximum training time. Since the first part of the model is performed by multiple devices in parallel, so the total time is determined by the slowest device. And we know each task is combined by many iterations. Denote $T_{i,j}$ as the j -th iteration of device d_i . We have

$$T_{total} = \max_{i \in N} \left\{ \sum_{j=1}^J T_{i,j} \right\}, \tag{1}$$

where J is the iteration frequency for the task. We want to ensure that after the whole training process, the learning accuracy will be larger than a threshold ϵ . We know that the iteration frequency have some relationship with the learning accuracy, but we do not know the exact formula between them. So we will first use a common function $f(J)$ to represent the relationship, then we have $f(J) \geq \epsilon$. In the next section, we will give a fitting formula for $f(J)$.

So for modeling the system model, we should first model $T_{i,j}$. To do that, define a binary variable $q(m_i^j)$ to indicate whether parameter aggregation is performed after the j -th iteration of device d_i . We have

$$q(m_i^j) = \begin{cases} 1 & \text{:parameter aggregation is required after this iteration;} \\ 0 & \text{:otherwise.} \end{cases} \tag{2}$$

Then $T_{i,j}$ can be expressed as

$$T_{i,j} = t_d^{f+b}(m_i^j) + t_{trans}^{f+b}(m_i^j) + t_s^{f+b}(m_i^j) + t_w(m_i^j) + q(m_i^j)t_{update}(m_i^j). \tag{3}$$

In Eq. (3), the time $T_{i,j}$ is made up of five parts. The first part $t_d^{f+b}(m_i^j)$ is the computing time on d_i for forward and backward propagation. The second part $t_{trans}^{f+b}(m_i^j)$ is the intermediate data transmission time between d_i and d_0 in the process of forward and backward propagation. The third part $t_s^{f+b}(m_i^j)$ is the

computing time on d_0 for the forward and backward propagation. The fourth part $t_w(m_i^j)$ is the waiting time for execution on d_0 . The fifth part $t_{update}(m_i^j)$ is the transmission time for parameter updating on d_0 . In the following we will give equations for these five parts.

For the first part $t_d^{f+b}(m_i^j)$, since we use model partition technique, define a binary variable $x_r(m_i^j)$ to indicate whether the partition point is on the r -th layer. We have

$$x_r(m_i^j) = \begin{cases} 1 & \text{:the partition point is the } r\text{-th layer;} \\ 0 & \text{:otherwise.} \end{cases} \quad (4)$$

Since there is only one partition point for each task, we have $\sum_{r=1}^v x_r(m_i^j) = 1$.

Then $t_d^{f+b}(m_i^j)$ can be represented as

$$t_d^{f+b}(m_i^j) = t_d^f(m_i^j) + t_d^b(m_i^j) = \sum_{r=1}^v (x_r(m_i^j) \cdot (L_r^f(m_i^j) + L_r^b(m_i^j))), \quad (5)$$

where $L_r^f(m_i^j)$ represents the time from 1 to r -th layer on the device for forward propagation, and $L_r^b(m_i^j)$ represents the time from r -th to 1 layer on the device for backward propagation.

For the second part $t_{trans}^{f+b}(m_i^j)$, we have

$$t_{trans}^{f+b}(m_i^j) = t_{trans}^f(m_i^j) + t_{trans}^b(m_i^j) = \sum_{r=1}^v (x_r(m_i^j) \cdot (\frac{G_r^f(m_i^j)}{\beta_i^j \cdot B} + \frac{G_{r+1}^b(m_i^j)}{B})), \quad (6)$$

where β_i^j is the bandwidth allocation ratio for d_i , B is the total bandwidth, $G_r^f(m_i^j)$ is the amount of data of intermediate result at layer l_r of m_i^j for forward propagation, and $G_{r+1}^b(m_i^j)$ is the amount of data of intermediate result at layer l_{r+1} of m_i^j for backward propagation.

For the third part $t_s^{f+b}(m_i^j)$, we have

$$t_s^{f+b}(m_i^j) = \sum_{r=1}^v (x_r(m_i^j) \cdot (S_{r+1}^f(m_i^j) + S_{r+1}^b(m_i^j))), \quad (7)$$

where $S_{r+1}^f(m_i^j)$ is the computing time for server to execute the forward propagation from layer l_{r+1} to l_v , and accordingly, $S_{r+1}^b(m_i^j)$ is the computing time for server to execute the backward propagation from l_v to l_{r+1} .

For the fourth part $t_w(m_i^j)$, since in our learning framework, multiple devices cooperate with edge server for training, and the limited resources of edge server, they cannot be processed at the same time. So there is a waiting queue. Let $\tau(m_i^j)$ represents the start time of the j -th iteration on d_i . Since there is no interval between iterations of the same device, we have

$$\tau(m_i^j) = \sum_{k=1}^{j-1} T_{i,k}. \quad (8)$$

Denote $AT(m_i^j)$ as the arrival time of m_i^j to d_0 , then we have

$$AT(m_i^j) = \tau(m_i^j) + t_d^f(m_i^j) + t_{trans}^f(m_i^j), \tag{9}$$

where $t_d^f(m_i^j)$ is the forward propagation time on the device and $t_{trans}^f(m_i^j)$ is the transmission time. We express the wait time recursively, with $Z(m_{i'}^{j'})$ representing the previous task of m_i^j . We have

$$Z(m_{i'}^{j'}) = \begin{cases} 1 & :m_{i'}^{j'} \text{ is the previous task of } m_i^j; \\ 0 & : \text{otherwise.} \end{cases} \tag{10}$$

Then $t_w(m_i^j)$ can be expressed as

$$t_w(m_i^j) = \max\{0, \sum_{i'=1}^N \sum_{j'=1}^j Z(m_{i'}^{j'}) \cdot (t_w(m_{i'}^{j'}) + t_s^{f+b}(m_{i'}^{j'}) - AT(m_{i'}^{j'}))\}. \tag{11}$$

For the fifth part $t_{update}(m_i^j)$, since the parameter aggregation will be done on the edge server, which means we need to transfer the parameter on the device to the edge server, and then the updated parameter should be returned to the device. So we have

$$t_{update}(m_i^j) = \sum_{r=1}^v (x_r(m_i^j) \cdot (\frac{H_r^b(m_i^j)}{\beta_i^j \cdot B} + \frac{H_r^b(m_i^j)}{B})), \tag{12}$$

where $H_r^b(m_i^j)$ is the parameter quantity size from l_1 to l_r . Therefore, the optimization problem is formulated as follows:

$$\begin{aligned} & \min T_{total}. \\ & \text{s.t. (3) (5) (6) (7) (11)} \\ & \sum_{r=1}^v x_r(m_i^j) = 1; \\ & 0 \leq \beta_i^j \leq 1; \\ & \epsilon \geq \epsilon^{min}. \end{aligned} \tag{13}$$

In (13), $q(m_i^j)$, $Z(m_{i'}^{j'})$ and other notations are all constants or determined values for specific network. $x_r(m_i^j)$ and β_i^j are variables. $x_r(m_i^j)$ is a binary variables, which almost appears in all items with different forms. β_i^j is continuous variables, and it appears in the denominator. Therefore, to solve this problem in polynomial time, we need to further analyze and find a way to reduce the complexity of the problem.

3 Algorithm

In this section, we introduce the proposed algorithm for the above optimization problem. Firstly, we give an expression for $f(J)$ by the fitting method from experimental data. Secondly, we obtain several optional partition points for each

device and reduce the range of $x_r(m_i^j)$ from the PPS algorithm in [22]. Finally, we propose an algorithm to dynamically adjust the partition point and bandwidth allocation for getting smaller training time.

3.1 Iterations Analysis

In this subsection, we will first discuss $f(J)$. In fact, the relationship between the learning accuracy $f(J)$ and the iteration frequency J is hard for modeling. Since many DNN inner features will affect the modeling, such as the characteristics of training data, the smoothness, the convexity of loss function and the characteristics of gradient, and so on. As we have mentioned in the related work, Some papers have done their work for finding the relationship, but these are not suit for the asynchronous iteration. So in our work, we try to model the relationship by means of experimental data fitting.

The DNN model used in our experiments is VGG-16, and the dataset for training is cifar-10. For each device, it will first do local iterations for some times by using the SGD, and then transmit parameters to the server for performing the parameter aggregation. In asynchronous iteration, a task performs parameter aggregation after local iterations without waiting for other tasks. Therefore, models that are more recently updated should have a larger weight in the aggregation. We have

$$\omega_{t+1}^s = (1 - \alpha)\omega_t^s + \alpha\omega_j^i; \quad (14)$$

where ω_t^s is the current global parameters on the server for t parameter aggregation, and ω_j^i is the local parameters for j local iteration. α is a parameter for the staleness weight, which can be calculated by $\alpha = a^{-(M-j)}$, where M is the current largest local iteration number for all the local models on the server and a is a constant.

Suppose the number of local iterations E is set to 5. According to the above settings and methods, our training results are shown in the Fig. 2, and the experimental data is fitted to obtain the formula (15).

$$f(J) = 77.52e^{(0.000324J)} - 92e^{(-0.0576J)}. \quad (15)$$

Suppose there are 10 devices involved in the training. Suppose the minimum accuracy $f_{min}(J)$ is 80%. Then by (15) we can get that it should take at least 100 iterations to achieve the accuracy. According to this, the following analysis and experiments are based on $J = 100$.

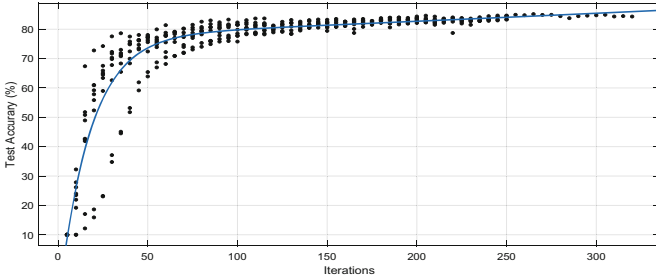


Fig. 2. The relationship between test accuracy and iterations

3.2 Problem Analysis

In this subsection, we continue to discuss our algorithm. In (13), $x_i(m_i^j)$ and β_i^j are dependent variables. In other words, $x_i(m_i^j)$ is the variable deciding the partition point, which will affect variable β_i^j . However, the solution spaces for $x_i(m_i^j)$ and β_i^j in the original problem are very large, so we need to find some way to reduce them. In our algorithm, we first use the PPS algorithm in [22] to select several optional partition points for $x_i(m_i^j)$. Then for β_i^j , we filter out some cases with smaller β_i^j by adjusting the partition point, and obtain the bandwidth allocation with smaller time. We give the detail discussion in the following.

We first discuss the $x_i(m_i^j)$. In fact, for a DNN model, not all layers are suitable for partition point since the output data of some layers are large. The larger l_r , the larger amount of uploaded parameters, which means if we choose a layer with large l_r for partition point, it will lead to high transmission cost. In our previous work, we have designed an algorithm named the PPS algorithm for selecting the optional partition points [22]. So in this paper, we first use the PPS algorithm and get c optional partition points for each device. Denoted the set of optional partition points for d_i as P_i .

Then for β_i^j , we know bandwidth allocation occurs when multiple tasks need to upload data simultaneously. For task m_i^j , which requires bandwidth in

$$\Phi(m_i^j) = [t_{im}^{st}, t_{im}^{en}] \cup [t_{pa}^{st}, t_{pa}^{en}], \tag{16}$$

where t_{im}^{st} , t_{im}^{en} are the start and end time of data transmission in the middle of forward propagation, they can be expressed as

$$t_{im}^{st} = \tau(m_i^j) + t_d^f(m_i^j), \tag{17}$$

$$t_{im}^{en} = t_{im}^{st} + t_{trans}^f(m_i^j). \tag{18}$$

And t_{pa}^{st} , t_{pa}^{en} are the start and end time of parameter upload on device d_i , they can be expressed as

$$t_{pa}^{st} = t_{im}^{en} + t_s^{f+b}(m_i^j) + t_{trans}^b(m_i^j) + t_d^b(m_i^j) + t_w(m_i^j), \tag{19}$$

$$t_{pa}^{en} = t_{pa}^{st} + t_{update}(m_i^j). \tag{20}$$

If task m_i^j need bandwidth in $\Phi(m_i^j)$, the bandwidth needs to be allocated. In general, more tasks competing for bandwidth will result in higher communication time. Therefore, we expect to reduce the number of tasks that generate bandwidth contention to reduce communication time.

Based on this, we design the Optimize Bandwidth Allocation (OBA) algorithm to dynamically adjust the partition point and bandwidth allocation. That is, reduce the tasks that will cause bandwidth competition to some extent when determining the corresponding partition points of tasks.

3.3 OBA Algorithm

According to the analysis in the previous section, the OBA algorithm can be summarized into the following three steps:

- 1) Firstly, a complete partition point allocation set P of partition points is obtained according to P_i , and the partition point allocation in P is calculated to obtain p_k with the minimum number of tasks requiring bandwidth, and β_i^j is calculated according to the partition points in p_k ;
- 2) Calculate the arrival time of tasks according to the partition point allocation p_k , and process tasks on server according to first come first service (FCFS);
- 3) For the next task to be processed, judge whether there are still optional partition points. If there are, calculate Φ in the partition point allocation including the remaining optional partition points, find the appropriate partition point allocation $p_{k'}$ and β_i^j to minimize the time. Otherwise, the original partition point allocation p_k and β_i^j are still used.

The step 3 will be repeated until the ending of training. The algorithm is shown in Algorithm 1. Line 1-13 are the process of training with finding the optimal partition point allocation in each iteration.

Algorithm 1. Optimize Bandwidth Allocation Algorithm

Input: P_i , the set of optimal partition points for each device; d_i , the i -th device; l_r , the partition point; j , the number of iteration; p_k , the initial partition point allocation;

Output: The minimum total time, T_{total} ;

```

1: for  $j = 2, 3, \dots, J$  do
2:   for  $d_i, l_r \in p_k$  do
3:     Compute  $AT(m_i^j)$  according to Eq.(9);
4:   end for
5:   Find  $m_i^{j'}$  whose arrival time is second only to that of the previous task  $m_i^j$  to
   start the next iteration;
6:   if  $m_i^{j'}$  still has optional partition points then
7:     Find the partition point allocation in  $P'$  including the remaining optional
   partition points, compute  $\beta_i^j$  and  $T_{i,j}$  in Eq. (3);
8:     Use partition point allocation  $p_{k'}$  and  $\beta_i^j$  that minimize  $T_{i,j}$ ;
9:   else
10:    Continue to use allocation  $p_k$ ;
11:   end if
12:   Compute the total time,  $T_{total}$ ;
13: end for

```

4 Simulation and Experiment

In this section, we introduce experiments and simulations. We use GPU to simulate the computing power of edge server, and use CPUs of different computers as edge devices. To get some constants for the Sect. 2, such as $L_r^f(m_i^j)$, $S_{r+1}^f(m_i^j)$, $G_r^f(m_i^j)$, $H_r^b(m_i^j)$ and so on, we train VGG-16 to get the execution time, output data and parameters amount for each layer. Firstly, the forward and backward propagation of the model is performed 100 times, then the execution time and output data amount for each layer are calculated on average, and the parameters amount for each layer of the model is obtained by using the tool in pytorch.

In Subsect. 4.1, we first obtain the above basic data, then calculate the total time of forward, back propagation, parameters upload and download at different partition points according to PPS algorithm, and finally get the optional partition points set P_i of d_i . In Subsect. 4.2, we run our algorithm under different system bandwidth and different number of devices. Use the following methods for comparison: 1) **No-Partitioned**, asynchronous federated learning without model partition; 2) **Fixed-Point**, allocate bandwidth on demand in the case of fixed partition point; 3) **Average-Bandwidth**, average bandwidth allocation in the same case as our algorithm.

4.1 Result of Partition Points

First of all, we obtain the time of each layer of the VGG-16 model for device and server execution, respectively, as shown in Fig. 3. When the total bandwidth $B = 10$ MB/s, according to the PPS algorithm, we can obtain the time cost of a task at different partition point when the waiting time is ignored. One of the result

is shown as Fig. 4, and we get four partition points. We have already included intermediate data transmission, model parameters upload and download time when calculating the time cost. Therefore, we believe that these four partition points are consistent with the structural characteristic and network condition of the current model.

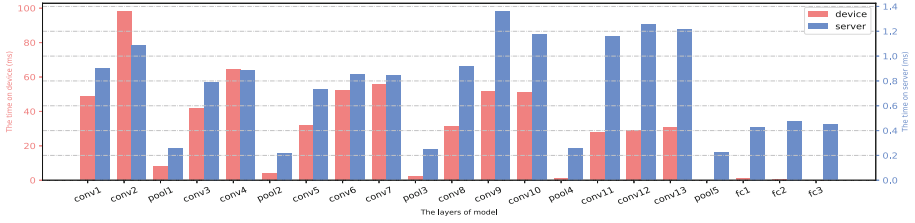


Fig. 3. The executing time on device and server

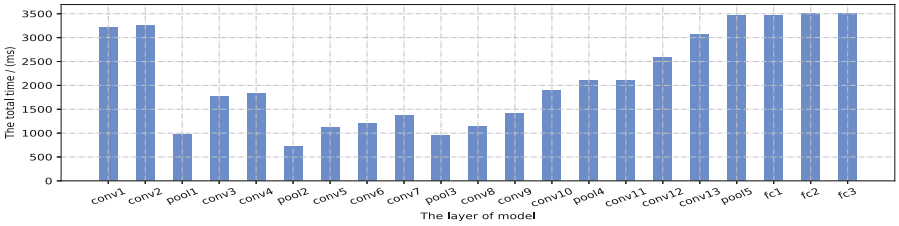


Fig. 4. The total time in different partition points

4.2 Algorithm Simulation

Firstly, we set the number of devices $N = 10$ and the iterations $J = 100$, then take the total bandwidth from 6 MB/s to 16 MB/s. The result is shown in Fig. 5a. As we can see, the higher the total bandwidth, the smaller the total time. In the case of small total bandwidth, our algorithm can achieve better result. Because the more limited bandwidth resources are, the more reasonable allocation is needed to play a full role. According to the experimental results, our algorithm can reduce the total training time by 60% on average compared with the No-Partitioned. The total time is reduced by 32% on average when compared with the Average-Bandwidth and the Fixed-Point.

In Fig. 5b, the total bandwidth is set to 10 MB/s, then we set different number of devices. It can be seen that with the increase of the number of devices, the training time also increases due to the increase of the number of iterations. Because there are more devices and data is spread across devices, asynchronous update requires more iterations to achieve the specified accuracy. Obviously, the effect of the proposed algorithm is always superior to other methods.

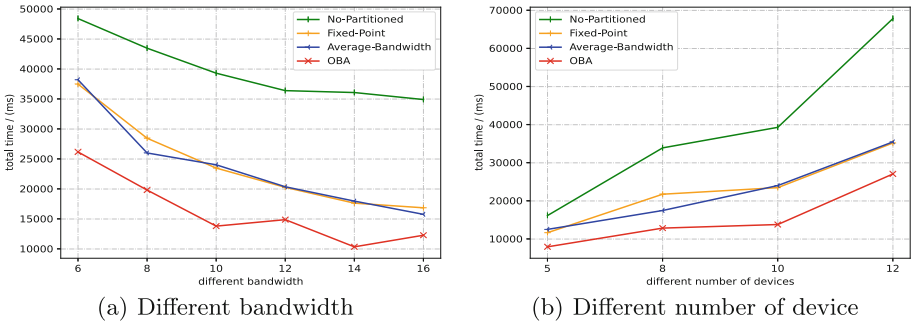


Fig. 5. Simulation result. (a) Total training time under different bandwidth. (b) Total training time under different number of device.

5 Conclusion

In this paper, we study asynchronous federated learning based on model partition in edge environment, aiming to ensure learning accuracy and reduce learning time. Firstly, we use the experimental method to get the expression of the relationship between learning accuracy and iteration frequency, and convert the learning accuracy to iteration frequency to express. Then, the time model of the learning system is established based on this, which contains a large number of variables related to model partition and bandwidth allocation, so it is difficult to solve directly. Therefore, we propose an algorithm to reduce the time as much as possible by dynamically adjusting model partition point and bandwidth allocation. Finally, the experimental results show that our algorithm can reduce the time by 32% to 60% compared with other methods.

References

1. McMahan, B., Moore, E., Ramage, D., Hampson, S., Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics vol. 54, pp. 1273–1282 (2017)
2. Liang, Y., Cai, Z., Yu, J., Han, Q., Li, Y.: Deep learning based inference of private information using embedded sensors in smart devices. *IEEE Network* **32**(4), 8–14 (2018). <https://doi.org/10.1109/MNET.2018.1700349>
3. Cai, Z., Xiong, Z., Xu, H., Wang, P., Li, W., Pan, Y.L.: Generative adversarial networks: a survey towards private and secure applications. *ACM Comput. Surv. (CSUR)* **54**(6), 1–38 (2021)
4. Mills, J., Hu, J., Min, G.: Communication-efficient federated learning for wireless edge intelligence in IoT. *IEEE Internet Things J.* **7**(7), 5986–5994 (2020)
5. Zhang, W., Gupta, S., Lian, X., Liu, J.: Staleness-aware Async-SGD for distributed deep learning. In: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, pp. 2350–2356 (2016)

6. Pang, J., Huang, Y., Xie, Z., Han, Q., Cai, Z.: Realizing the heterogeneity: a self-organized federated learning framework for IoT. *IEEE Internet Things J.* **8**(5), 3088–3098 (2021)
7. Xiong, Z., Cai, Z., Takabi, D., Li, W.: Privacy threat and defense for federated learning with non-i.i.d. data in AloT. *IEEE Trans. Industr. Inform.* **18**(2), 1310–1321 (2022)
8. Luo, S., Chen, X., Wu, Q., Zhou, Z., Yu, S.: HFEL: joint edge association and resource allocation for cost-efficient hierarchical federated edge learning. *IEEE Trans. Wireless Commun.* **19**(10), 6535–6548 (2020). <https://doi.org/10.1109/TWC.2020.3003744>
9. Yang, Z., Chen, M., Saad, W., Hong, C.S., Shikh-Bahaei, M.: Energy efficient federated learning over wireless communication networks. *IEEE Trans. Wireless Commun.* **20**(3), 1935–1949 (2021)
10. Li, X., Huang, K., Yang, W., Wang, S., Zhang, Z.: On the convergence of FedAvg on non-IID data. In: *International Conference on Learning Representations* (2020)
11. Chen, M., Poor, H.V., Saad, W., Cui, S.: Convergence time optimization for federated learning over wireless networks. *IEEE Trans. Wireless Commun.* **20**(4), 2457–2471 (2021)
12. Xu, J., Wang, H.: Client selection and bandwidth allocation in wireless federated learning networks: a long-term perspective. *IEEE Trans. Wireless Commun.* **20**(2), 1188–1200 (2021)
13. Amiri, M.M., Gündüz, D., Kulkarni, S.R., Poor, H.V.: Convergence of update aware device scheduling for federated learning at the wireless edge. *IEEE Trans. Wireless Commun.* **20**(6), 3643–3658 (2021)
14. Chen, M., Mao, B., Ma, T.: FedSA: a staleness-aware asynchronous federated learning algorithm with non-IID data. *Futur. Gener. Comput. Syst.* **120**, 1–12 (2021)
15. Lee, H.S., Lee, J.W.: Adaptive transmission scheduling in wireless networks for asynchronous federated learning. *IEEE J. Sel. Areas Commun.* **39**(12), 3673–3687 (2021)
16. Cai, Z., Shi, T.: Distributed query processing in the edge-assisted IoT data monitoring system. *IEEE Internet Things J.* **8**(16), 12679–12693 (2021)
17. Zhu, T., Shi, T., Li, J., Cai, Z., Zhou, X.: Task scheduling in deadline-aware mobile edge computing systems. *IEEE Internet Things J.* **6**(3), 4854–4866 (2019)
18. Zhu, G., Liu, D., Du, Y., You, C., Zhang, J., Huang, K.: Toward an intelligent edge: wireless communication meets machine learning. *IEEE Commun. Mag.* **58**(1), 19–25 (2020)
19. Chen, M., Semiari, O., Saad, W., Liu, X., Yin, C.: Federated echo state learning for minimizing breaks in presence in wireless virtual reality networks. *IEEE Trans. Wireless Commun.* **19**(1), 177–191 (2020)
20. Kang, Y., et al.: Neurosurgeon: collaborative intelligence between the cloud and mobile edge. *SIGPLAN Not.* **52**(4), 615–629 (2017)
21. Zhang, L., Xu, J.: Learning the optimal partition for collaborative DNN training with privacy requirements. *IEEE Internet Things J.* **8**, 1–11 (2021)
22. Shi, L., Xu, Z., Sun, Y., Shi, Y., Fan, Y., Ding, X.: A DNN inference acceleration algorithm combining model partition and task allocation in heterogeneous edge computing system. *Peer-to-Peer Netw. Appl.* **14**, 4031–4045 (2021)



QoE and Reliability-Aware Task Scheduling for Multi-user Mobile-Edge Computing

Weiming Jiang¹, Junlong Zhou^{1(✉)}, Peijin Cong¹, Gongxuan Zhang¹, and Shiyan Hu²

¹ School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China

{jwm, jlzhou, pjcong, gongxuan}@njjust.edu.cn

² School of Electronics and Computer Science, University of Southampton, Southampton, UK
S.Hu@soton.ac.uk

Abstract. Mobile-edge computing (MEC) has become a popular research topic from both academia and industry since it can alleviate the computation and power limitations of mobile devices by offloading computation-intensive and energy-consuming tasks from mobile users to nearby edge servers for remote execution. Existing papers have studied related problems, however, none of them considers the reliability of MEC systems that may suffer soft errors during execution and bit errors during offloading. In this work, we study the task offloading and scheduling problem targeting to maximize the quality of experience (QoE) of multi-user MEC systems under a certain reliability requirement. We propose to decompose the original problem into i) a task offloading optimization problem, ii) a task-to-server assignment problem for ensuring system reliability constraint, and iii) a computing resource allocation problem for maximizing system QoE. To address these sub-problems, we first obtain the optimal offloading decision using the discrete particle swarm optimization method. We then propose a reliability-optimality analysis-based task assignment heuristic and a utility-optimal resource allocation algorithm. Simulation results show that our scheme outperforms two state-of-the-art approaches and two baseline methods. The average improvement on QoE (quantified by offloading utility) achieved by our scheme is up to 63.2% under reliability requirement.

Keywords: MEC · QoE · Reliability · Scheduling · Resource allocation

1 Introduction

With the advancement of Internet of Things (IoT) technologies, various mobile applications such as autonomous driving and smart city emerge and are gradu-

Junlong Zhou is the corresponding author

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022

L. Wang et al. (Eds.): WASA 2022, LNCS 13473, pp. 380–392, 2022.

https://doi.org/10.1007/978-3-031-19211-1_32

ally changing people's daily lives [1]. The emerging mobile applications typically have a heavy demand of computation capability and energy of mobile devices for processing massive IoT data. However, mobile devices generally cannot meet this demand due to their limited computing resources and finite battery energy. Recently, a promising technology named mobile edge computing (MEC) [2] emerges to solve this issue by allowing mobile devices to offload applications to nearby edge servers for remote execution. Benefiting from MEC, mobile devices' computing ability and energy efficiency can be both effectively improved.

The key of MEC is *offloading* and *scheduling* that decide which task should be offloaded to which edge server for remote execution with what resource, to meet diverse performance requirements. Offloading and scheduling strategies for improving the quality of experience (QoE) of users in the MEC system have been studied from many aspects. For single-user MEC systems, Kuang et al. [3] propose a partial offloading and power allocation scheme to minimize latency and energy consumption of independent tasks. Wang et al. [4] develop a meta reinforcement learning based offloading method for dependent mobile applications to reduce latency. In multi-user MEC systems, the task offloading and scheduling problem becomes more challenging due to the competition among multiple users for resources. A dynamic resource allocation and computation offloading strategy [5] is presented to minimize system timing and energy cost. Gupta et al. [6] and Tran et al. [7] solve the joint task offloading and resource allocation problems that maximize the lifetime of multi-user MEC networks and optimize the offloading performance of multi-user MEC systems, respectively.

Although the above-mentioned approaches are effective in improving QoE performance, none of them considers reliability which is actually very important to the MEC systems for ensuring successful task offloading and execution. In an MEC system, no matter whether tasks are executed locally on the mobile devices or remotely on the edge servers, they all may suffer soft errors induced by transient faults. Besides, the tasks to be offloaded may suffer bit errors during the data transmission. Thus, it is highly necessary to enhance the reliability of MEC systems. In this paper, we study the task offloading and scheduling problem to maximize QoE of multi-user multi-server MEC systems while satisfying the reliability requirement. Our major contributions are listed as follows.

1. We consider both transmission reliability and execution reliability of MEC systems and provide a reliability-optimality analysis on the task-to-server assignment. Based on the analysis, we propose a theorem and then develop a task assignment heuristic following the theorem.
2. We formulate the problem of allocating computing resources to tasks offloaded on edge servers for maximizing utility. We prove that the utility aware resource allocation problem is a convex optimization problem when task assignment have been decided, we use Karush-Kuhn-Tucker (KKT) conditions to solve it.
3. We implement extensive simulations to evaluate the performance of the proposed scheme in improving utility and reliability. The simulation results show the efficacy of the proposed scheme.

2 System Model

2.1 MEC System Model

Consider a multi-user multi-server MEC system shown in Fig. 1. In such an MEC system, a base station is equipped with multiple edge servers. Note that an edge server can be either a physical server or a virtual machine with moderate computational capability. The base station can help mobile users complete computation-intensive tasks by offloading these tasks to edge servers via the wireless channel. Let $U = \{U_1, \dots, U_N\}$ and $S = \{S_1, \dots, S_M\}$ denote the sets of N mobile users and M edge servers in the MEC system. Similar to [7], we assume that each mobile user has one computation task to execute at a time. The tasks of mobile users are all atomic and independent, and cannot be divided into sub-tasks. Let $\Gamma = \{\tau_1, \dots, \tau_N\}$ be the set of tasks to be executed by N users. The task of user U_i ($1 \leq i \leq N$), represented by τ_i , is characterized by a triple $\{d_i, c_i, \nu_i\}$ where d_i is the amount of data transferred for offloading, c_i is the number of computation cycles to be completed, and ν_i is the task vulnerability factor when suffering transient faults [8].

Each task could be executed either locally or offloaded to an edge server. Let $X = \{x_i | 1 \leq i \leq N\}$ be the task offloading decision where $x_i = 1$ when task τ_i is offloaded to an edge server and $x_i = 0$ otherwise. Given an offloading decision, task set Γ can be divided into two sets: local task set Γ_{loc} and offloaded task set Γ_{off} . The MEC system allows multiple users to offload their tasks to the same server concurrently by sharing the server's computing resource. When a task is offloaded to an edge server, the server will allocate its computing resource to execute the task and return the result to the user. The computing resource of server S_j ($1 \leq j \leq M$) is quantified by its computing capacity F_j (in cycles/s) [7]. Let $F = \{f_{ij} | 1 \leq i \leq N, 1 \leq j \leq M\}$ denote the computing resource allocation policy where f_{ij} refers to the computing resource allocated by server S_j to task τ_i and it holds for $f_{ij} \leq F_j$.

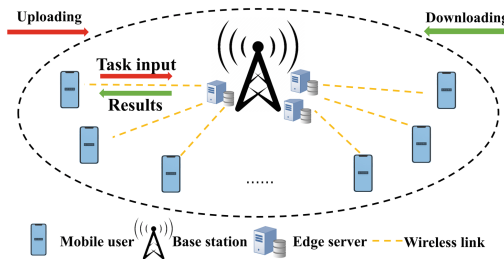


Fig. 1. A multi-user multi-server MEC system.

The orthogonal frequency division multiple access technology [9] is adopted in the uplink to support multiple access to edge servers. The base station's

bandwidth B is divided into equal sub-bands and each user is assigned one sub-band. According to the Shannon-Hartley theorem [10], the data uploading rate r_i from user U_i to the base station is $r_i = \frac{B}{n} \log_2 \left(1 + \frac{p_i g_i}{\omega + \sum_{k=1, k \neq i}^n p_k g_k} \right)$, where p_i is transmission power, g_i is channel gain, and ω is background noise. Same as in [7], the time of transferring output result is not considered.

2.2 Computation Model

When a mobile user offloads its task to an edge server, the device would save its energy for execution but take extra time and energy for data transmission [3–7].

Local Computing. If task τ_i is executed locally on user device U_i , the local computing latency t_i^{loc} is derived as

$$t_i^{\text{loc}} = \frac{c_i}{f_i^{\text{loc}}} \quad (1)$$

where f_i^{loc} is the computing capability (in cycles/s) of device U_i . The energy E_i^{loc} consumed by locally executing task τ_i is

$$E_i^{\text{loc}} = \kappa (f_i^{\text{loc}})^2 c_i \quad (2)$$

where κ is the processor-dependent coefficient [10].

Offloading Computing. If task τ_i is executed remotely, it needs to be offloaded to an edge server first and then be executed by the server. Thus, the time cost t_i^{off} is the sum of transmission delay t_i^{trans} and execution delay t_i^{exe} ,

$$t_i^{\text{off}} = t_i^{\text{trans}} + t_i^{\text{exe}} = \frac{d_i}{r_i} + \frac{c_i}{f_{ij}}. \quad (3)$$

The energy consumed by device U_i for data transmission is

$$E_i^{\text{trans}} = p_i \times t_i^{\text{trans}} \quad (4)$$

2.3 Reliability Model

Transmission Reliability. During the transmission, tasks may suffer from noise and interference bit errors, as well as bit synchronization errors over transmission links [11]. Following the bit error model [11], we can derive the transmission reliability of task τ_i as

$$\mathcal{R}_i^{\text{trans}} = e^{-\gamma \times t_i^{\text{trans}}} \quad (5)$$

where γ is the constant bit error rate.

Execution Reliability. During the execution, transient faults may induce soft errors, which appear in a short period of time and disappear without damaging hardware [12]. Following the exponential distribution [13], the execution reliability of task τ_i is expressed as

$$\mathcal{R}_i^{\text{exe}} = e^{-\lambda(f_i) \times \nu_i \times \frac{c_i}{f_i}} \quad (6)$$

where $\lambda(f_i)$ is the raw fault rate when executing task τ_i at processor frequency f_i and ν_i is the vulnerability factor of task τ_i . $f_i = f_i^{\text{loc}}$ if task τ_i is executed locally and $f_i = f_{ij}$ otherwise.

System Reliability. Given the transmission reliability and execution reliability, the task reliability is formulated as

$$\mathcal{R}_i = \begin{cases} \mathcal{R}_i^{\text{exe}}, & \text{if } x_i = 0 \\ \mathcal{R}_i^{\text{trans}} \times \mathcal{R}_i^{\text{exe}}, & \text{if } x_i = 1 \end{cases} \quad (7)$$

where x_i indicates whether task τ_i is offloaded. Obviously, system reliability depends on the successful transmission and execution of all independent tasks. Therefore, the system reliability can be readily derived as

$$\mathcal{R}_{\text{sys}} = \prod_{\tau_i \in \Gamma} \mathcal{R}_i. \quad (8)$$

3 Problem Formulation and Methodology Overview

3.1 Problem Definition

Task execution latency and device battery energy are two main factors affecting the QoE of users in the MEC system. To quantify the users' QoE of offloading, we define a metric *offloading utility* that reflects the improvement in execution latency and energy efficiency achieved by offloading. The offloading utility of device U_i is formulated as

$$\mathcal{U}_i = x_i \left(\alpha_i \left(\frac{t_i^{\text{loc}} - t_i^{\text{off}}}{t_i^{\text{loc}}} \right) + \beta_i \left(\frac{E_i^{\text{loc}} - E_i^{\text{trans}}}{E_i^{\text{loc}}} \right) \right) \quad (9)$$

where α_i and β_i indicate the user's preference on latency and energy respectively, and they hold for $\alpha_i + \beta_i = 1$, $\alpha_i, \beta_i \in [0, 1]$. We aim to maximize the offloading utility of multi-user multi-server MEC systems under the reliability constraint. The offloading utility optimization problem is formulated as

$$\max: \quad \sum_{i=1}^N \mathcal{U}_i \quad (10)$$

$$\text{s.t.} \quad x_i \in \{0, 1\}, \quad \forall i = 1, \dots, N \quad (11)$$

$$f_{ij} > 0, \quad \forall \tau_i \in \Gamma_{\text{off}}, \quad \forall j = 1, \dots, M \quad (12)$$

$$\sum_{\tau_i \in \Gamma_j} f_{ij} \leq F_j, \quad \forall j = 1, \dots, M \quad (13)$$

$$\mathcal{R}_{\text{sys}} \geq \mathcal{R}_{\text{th}} \quad (14)$$

Eq. (11) implies that each user can only execute its task either locally or remotely. Equations (12) and (13) require that each offloaded task should be allocated a certain amount of computing resources within the capacity of its offloaded server, where $\Gamma_j \subset \Gamma_{\text{off}}$ is the set of tasks offloaded to server S_j . Equation (14) ensures that system reliability \mathcal{R}_{sys} cannot be lower than a threshold \mathcal{R}_{th} .

3.2 Overview

The studied problem involves the decisions on i) the tasks to be offloaded, ii) the assignment of offloaded tasks to edge servers, and iii) the allocation of computing resources to tasks assigned on servers. Considering the combinatorial nature of this problem, we propose to decompose the original problem into three sub-problems. Since many papers have studied offloading decision optimization problem, we derive the offloading decision X which is a 0–1 sequence using the discrete particle swarm optimization method [14]. Then we can focus on other two sub-problems. Given an offloading decision X that divides task set Γ into two sub-sets Γ_{loc} and Γ_{off} , we propose a reliability-aware task assignment approach that judiciously assigns the tasks in set Γ_{off} to edge servers for increasing system reliability. After that, we formulate the computing resource allocation problem F as a convex optimization problem with given task offloading and assignment strategies $\{X, A\}$. We solve the convex problem to derive the resource allocation solution F using Karush-Kuhn-Tucker (KKT) conditions.

4 Reliability-Aware Task Assignment

Fault-tolerant techniques such as checkpointing and replication are effective in improving reliability but they are both time/energy-consuming. This paper proposes a task-to-server assignment scheme that exploits the heterogeneity of tasks and servers to increase system reliability without incurring extra time/energy overhead.

Motivated by [13], we design our task assignment scheme which is however quite different from the approach in [13] from many aspects such as the formulation of system reliability and the implementation of heuristic algorithm. Given a task offloading decision X , we can readily derive the local task set Γ_{loc} and the offloaded task set Γ_{off} . Then, the system reliability given in Eq. (8) can be re-written as

$$\begin{aligned} \mathcal{R}_{\text{sys}} &= \prod_{\tau_i \in \Gamma} \mathcal{R}_i = \prod_{\tau_i \in \Gamma_{\text{loc}}} \mathcal{R}_i \times \prod_{\tau_i \in \Gamma_{\text{off}}} (\mathcal{R}_i^{\text{trans}} \times \mathcal{R}_i^{\text{exe}}) \\ &= e \left(- \sum_{\tau_i \in \Gamma_{\text{loc}}} \lambda (f_i^{\text{loc}}) \times \nu_i \times \frac{c_i}{f_i^{\text{loc}}} \right) \times e \left(- \sum_{\tau_i \in \Gamma_{\text{off}}} \gamma \times \frac{d_i}{r_i} \right) \\ &\quad \times e \left(- \sum_{\tau_i \in \Gamma_{\text{off}}} \lambda (f_{ij}) \times \nu_i \times \frac{c_i}{f_{ij}} \right). \end{aligned} \quad (15)$$

The first item $\prod_{\tau_i \in \Gamma_{\text{loc}}} \mathcal{R}_i$ and second item $\prod_{\tau_i \in \Gamma_{\text{off}}} \mathcal{R}_i^{\text{trans}}$ are constant since all parameters are constant or related to given tasks or devices. Thus we can

conclude that \mathcal{R}_{sys} is determined by the third item for a given offloading decision X and it is maximized when the third item reaches its maximum value. In other words, maximizing \mathcal{R}_{sys} is equivalent to maximizing $\prod_{\tau_i \in \Gamma_{\text{off}}} \mathcal{R}_i^{\text{exe}}$.

Consider a case that each task τ_i in set Γ_{off} is executed at its offloaded server's maximum capacity F_j . The corresponding reliability is expressed as

$$\begin{aligned} \prod_{\tau_i \in \Gamma_{\text{off}}} \mathcal{R}_i^{\text{exe}}(F_j) &= e^{\left(-\sum_{j=1}^M \frac{\lambda(F_j)}{F_j} \times (\sum_{\tau_i \in \Gamma_j} \nu_i \times c_i)\right)} \\ &= e^{-(y_1 \times z_1 + \dots + y_M \times z_M)} = e^{-\mathbf{y} \cdot \mathbf{z}} \end{aligned} \tag{16}$$

where Γ_j is the set of tasks offloaded to server S_j . Through the above algebra transformation, the reliability is determined by the product of two vectors \mathbf{y} and \mathbf{z} where $y_j = \frac{\lambda(F_j)}{F_j} \in \mathbf{y}$ represents the vulnerability index of server S_j and $z_j = \sum_{\tau_i \in \Gamma_j} \nu_i \times c_i \in \mathbf{z}$ represents the vulnerability index of task set Γ_j . We can easily find that the two vectors have the following characteristics with respect to reliability. ❶ For a given offloaded task set Γ_{off} , the sum of $\nu_i \times c_i$ of all tasks is a constant, i.e., $z_1 + \dots + z_M$ is a constant. ❷ A task/server with a relatively smaller vulnerability index means that it is relatively more reliable. ❸ \mathbf{y} is fixed for a given MEC system while \mathbf{z} is not and depends on task assignment. Based on the three characteristics, we propose a theorem (i.e., **Theorem 1**) on task assignment that derives the minimal $\mathbf{y} \cdot \mathbf{z}$ to maximize $\prod_{\tau_i \in \Gamma_{\text{off}}} \mathcal{R}_i^{\text{exe}}$ and hence \mathcal{R}_{sys} . Although we assume $f_{ij} = F_j$ in the above analysis, we can still derive the three characteristics and hence deduce **Theorem 1** for any other cases of f_{ij} once f_{ij} has been determined. The proof of the theorem is omitted due to page limit.

Algorithm 1: Reliability-Aware Task Assignment

Input: Γ_{off} and $S = \{S_1, \dots, S_M\}$ meeting $y_1 \leq y_2 \leq \dots \leq y_M$
Output: Task assignment strategy $A = \{\Gamma_1, \dots, \Gamma_M\}$

```

1 for  $j = 1$  to  $M$  do
2    $\Gamma_j = \emptyset$ ;
3 for  $i = 1$  to  $\text{sizeof}(\Gamma_{\text{off}})$  do
4   sort tasks in  $\Gamma_{\text{off}}$  in descending order by  $\nu_i \times c_i$ .
5    $j = 1$  and  $i = 1$ ;
6 while  $i \leq \text{sizeof}(\Gamma_{\text{off}})$  do
7    $\Gamma_j = \Gamma_j + \tau_i, i++, j++$ ;
8   if  $j > M$  then
9      $j = 1$ ;

```

Theorem 1. *Given an offloading strategy X , if the server vulnerability index y_i in \mathbf{y} satisfy $y_1 \leq \dots \leq y_M$ and the sum of task set vulnerability index z_j in \mathbf{z} is constant, then $\mathbf{y} \cdot \mathbf{z}$ is minimized (system reliability \mathcal{R}_{sys} is maximized) if $z_1 \geq \dots \geq z_M$ holds.*

Motivated by **Theorem 1**, we design a heuristic algorithm (i.e., **Algorithm 1**) that assigns the unreliable tasks with relatively larger vulnerability to the reliable servers with relatively smaller vulnerability index.

5 Utility-Optimal Resource Allocation

After deriving the task assignment strategy, we formulate and solve the problem of allocating resources to all tasks assigned on edge servers for maximizing offloading utility.

Substituting Eq. (9) into Eq. (10), we have

$$\begin{aligned} \sum_{\tau_i \in \Gamma_{\text{off}}} \mathcal{U}_i &= \sum_{j=1}^M \sum_{\tau_i \in \Gamma_j} \left(1 - \frac{\alpha_i d_i f_i^{\text{loc}}}{r_i c_i} - \frac{\beta_i p_i d_i}{r_i \kappa (f_i^{\text{loc}})^2 c_i} \right) \\ &\quad - \sum_{j=1}^M \sum_{\tau_i \in \Gamma_j} \frac{\alpha_i f_i^{\text{loc}}}{f_{ij}} \end{aligned} \quad (17)$$

where the transmit power p_i and transmission rate r_i of user device U_i are both known after task assignment. Thus, the first item of Eq. (17) is a constant and then the original problem given in Eqs. (10)-(14) can be reduced to

$$\min: \quad \varphi(f_{ij}) = \sum_{j=1}^M \sum_{\tau_i \in \Gamma_j} \frac{\alpha_i f_i^{\text{loc}}}{f_{ij}} \quad (18)$$

$$\text{s.t.}: \quad \sum_{\tau_i \in \Gamma_j} f_{ij} \leq F_j, \quad \forall j = 1, \dots, M \quad (19)$$

$$f_{ij} > 0, \quad \forall \tau_i \in \Gamma_j, \quad \forall j = 1, \dots, M \quad (20)$$

where the reliability constraint is not considered since it has most probably been satisfied via our reliability-aware task assignment and can be checked after resource allocation. Clearly, the reduced problem is a function of f_{ij} which represents the computing resource allocated by server S_j to task τ_i . Our goal is to find an optimal allocation strategy F to minimize the value of function $\varphi(f_{ij})$. Below, we propose a theorem to show that $\varphi(f_{ij})$ is a convex function in its domain. We omit the proof due to page limit.

Theorem 2. *The computing resource allocation problem formulated in Eqs. 18–20 is a convex optimization problem.*

To solve the convex optimization problem, we build the Lagrangian function of Eqs. (18)–(20) and then apply the Karush-Kuhn-Tucker (KKT) conditions to solve the problem. The Lagrangian function is defined as

$$L(\varphi(f_{ij}), v) = \sum_{j=1}^M \sum_{\tau_i \in \Gamma_j} \frac{\alpha_i f_i^{\text{loc}}}{f_{ij}} + \sum_{j=1}^M v_j \left(\sum_{\tau_i \in \Gamma_j} f_{ij} - F_j \right) \quad (21)$$

where $v = [v_1, v_2, \dots, v_M]$ is the Lagrange multiplier vector.

The KKT conditions are given as follows.

$$\frac{\partial L(\varphi(f_{ij}), v)}{\partial f_{ij}} = 0, \forall \tau_i \in \Gamma_{\text{off}}, 1 \leq j \leq M \tag{22}$$

$$\sum_{\tau_i \in \Gamma_j} f_{ij} - F_j \leq 0, \forall \tau_i \in \Gamma_{\text{off}}, 1 \leq j \leq M \tag{23}$$

$$v_j \geq 0, \forall \tau_i \in \Gamma_{\text{off}}, 1 \leq j \leq M \tag{24}$$

$$v_j \left(\sum_{\tau_i \in \Gamma_j} f_{ij} - F_j \right) = 0, \forall \tau_i \in \Gamma_{\text{off}}, 1 \leq j \leq M \tag{25}$$

Finally, we use KKT conditions to solve the convex problem and obtain the optimal resource allocation solution as

$$f_{ij}^* = \frac{F_j \sqrt{\alpha_i f_i^{\text{loc}}}}{\sum_{\tau_i \in \Gamma_j} \sqrt{\alpha_i f_i^{\text{loc}}}}, \forall \tau_i \in \Gamma_j, 1 \leq j \leq M \tag{26}$$

We summarize the basic steps of our method to derive the utility-optimal resource allocation strategy $F^* = \{f_{ij}^* | \forall \tau_i \in \Gamma_j, \forall j = 1, \dots, M\}$ in **Algorithm 2**.

Algorithm 2: Utility-Optimal Resource Allocation

Input: Task assignment strategy $A = \{\Gamma_1, \dots, \Gamma_M\}$

Output: Resource allocation strategy $F^* = \{f_{ij}^* | \forall \tau_i \in \Gamma_j, \forall j = 1, \dots, M\}$

- 1 **for** $j = 1$ to M **do**
 - 2 **for** $i = 1$ to $\text{sizeof}(\Gamma_j)$ **do**
 - 3 calculate optimal f_{ij}^* by Eq. (26) and set $f_{ij} = f_{ij}^*$;
 - 4 Check whether the system reliability \mathcal{R}_{sys} reaches the threshold \mathcal{R}_{th} .
-

6 Simulation

This section validates our proposed scheme through extensive simulation experiments. In the experiments, we consider three simulated MEC systems that consist of 20 mobile devices ($N = 20$), 35 mobile devices ($N = 35$), and 50 mobile devices ($N = 50$), respectively. The computing capability f_i^{loc} of mobile devices is varied in the range of $[1.0, 1.2]GHz$. Regarding the mobile user preference on latency and energy in QoE, we set them as equal by $\alpha_i = \beta_i = 0.5$. Each simulated MEC system is equipped with a base station and three edge servers to provide offloading services to mobile devices. We use three resource settings (i.e., low-resource case, medium-resource case, and high-resource case) to evaluate the performance of our proposed scheme and comparative algorithms. In the low-resource case, the base station’s bandwidth B is set to 15MHz and the computing capacity F_j of three edge servers is set to $\{8GHz, 10GHz, 12GHz\}$. In the

medium-resource case, the base station's bandwidth B is set to 25MHz and the computing capacity F_j of three edge servers is set to $\{14\text{GHz}, 16\text{GHz}, 18\text{GHz}\}$. In the high-resource case, the base station's bandwidth B is set to 35MHz and the computing capacity F_j of three edge servers is set to $\{20\text{GHz}, 22\text{GHz}, 24\text{GHz}\}$. The background noise ω is set to -100dBm . The transmission power of mobile devices is uniformly distributed within the range of $[0.1, 0.5]\text{W}$. The energy coefficient κ is set to 5×10^{-27} [7]. For each task τ_i , its data size d_i , number of execution cycles c_i , and vulnerability factor ν_i are uniformly distributed in the range of $[200, 1000]\text{KB}$, $[500, 2500]\text{Megacycles}$, and $[0, 1]$, respectively.

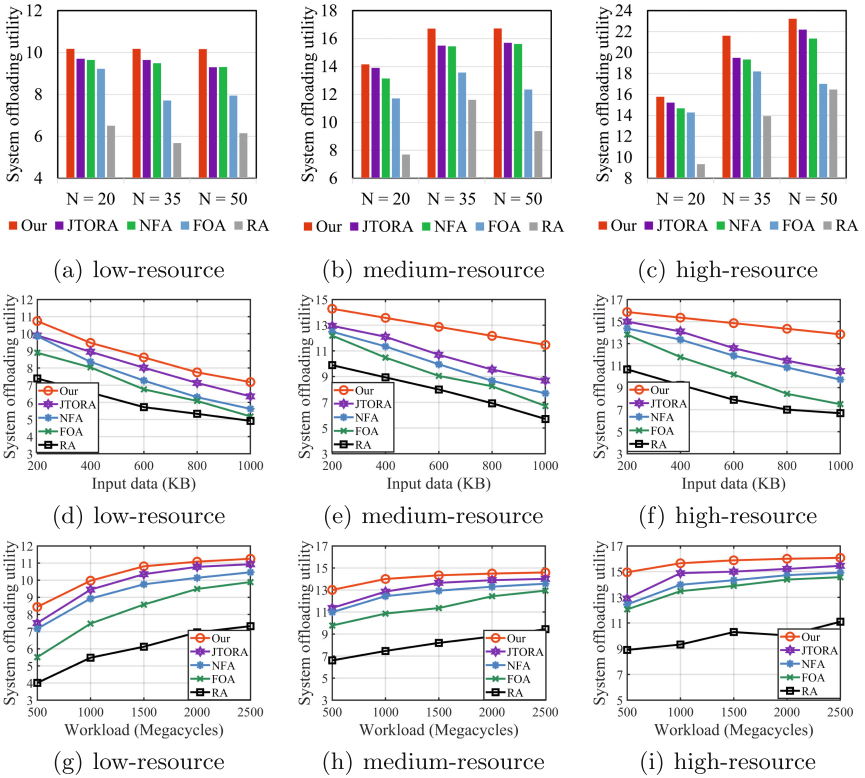


Fig. 2. Offloading utility with varying (a)-(c) number of mobile devices (d)-(f) task input data and (g)-(i) workloads.

We compare our scheme with two state-of-the-art approaches JTORA [7] and NFA [15] and baseline methods FOA and RA that are all applied to solve our problem, with respect to QoE (quantified by offloading utility) and reliability.

- JTORA [7] is a Joint Task Offloading and Resource Allocation algorithm that uses mixed integer nonlinear programming and decomposition methods to find near-optimal solution.

- NFA [15] is a Novel Firefly Algorithm that develops a mapping operator and a composite heuristic to find the optimal solution efficiently.
- FOA is a Full-Offload Algorithm that offloads all tasks of mobile devices to edge servers for execution.
- RA is a Random Algorithm that randomly decides offloading and scheduling.

Figure 2(a)-(c) compares the offloading utility achieved by our scheme and JTORA, NFA, FOA, RA under varying resource settings and number of mobile devices. The results clearly indicate that our scheme always outperforms JTORA, NFA, FOA, and RA in increasing offloading utility regardless of resource settings and the number of mobile devices. Compared to JTORA, NFA, FOA, and RA, the average improvement on offloading utility realized by our scheme are 6.1%, 8.1%, 23.8% and 63.2%, respectively. The maximum improvement achieved by our scheme can be up to 84.0% when compared to RA, in the medium-resource case of Fig. 2(b) with $N = 20$. As shown in the three cases of Fig. 2(a)-(c), we can find that offloading utility increases along with the growth in resource capacity. This is because that more bandwidth and computing resources are provided, more tasks can be offloaded and hence higher system offloading utility can be derived.

For a comprehensive evaluation, Fig. 2(d)-(f) and (g)-(i) demonstrate offloading utility achieved by our scheme and JTORA, NFA, FOA, RA under varying task input data and workloads, respectively. The results show that our scheme always has the maximum offloading utility among the five methods regardless of task data size and workload. The maximum utility improvement achieved by our scheme can be up to 104.6% when compared to RA, in the high-resource case of Fig. 2(f) with $d_i = 800$. In addition, from the Fig. 2(d)-(f) and (g)-(i) we find that offloading utility decreases with the increase of task data size d_i while it increases with the growth of task workload c_i . This implies that tasks with smaller data size and larger workload are more suitable for offloading and remote execution on servers.

We also compare the system reliability achieved by our scheme and JTORA, NFA, FOA, RA under varying number of mobile devices, task input data and task workloads, respectively. Due to page limit, we only present the system reliability under varying number of mobile devices. As shown in Table 1, unlike the other four approaches, our scheme can always maintain the system reliability at a high level and thus more probably meets the reliability constraint. For example, in the low-resource case, the system reliability realized by our scheme can be up to 12.9%, 20.7%, 23.4%, and 32.8% higher than that of NFA, JTORA, FOA, and RA, respectively. The same observation can be found in the cases of varying task input data and task workloads.

Table 1. System reliability with varying number of devices.

	Low-resource			Medium-resource			High-resource		
	N = 20	N = 35	N = 50	N = 20	N = 35	N = 50	N = 20	N = 35	N = 50
NFA	0.949	0.927	0.847	0.964	0.912	0.870	0.974	0.932	0.889
JTORA	0.940	0.833	0.792	0.962	0.899	0.805	0.974	0.921	0.865
FOA	0.938	0.831	0.775	0.962	0.890	0.795	0.972	0.919	0.841
RA	0.913	0.822	0.720	0.938	0.840	0.786	0.951	0.861	0.822
Our	0.976	0.966	0.956	0.979	0.946	0.928	0.983	0.945	0.910

7 Conclusion

To solve the task scheduling issues in multi-user multi-server MEC systems, this paper proposes a task assignment heuristic that increases system reliability based on the reliability-optimality analysis and designs a convex optimization method to perform computing resource allocation to maximize system QoE (quantified by offloading utility). Experiment results show the effectiveness of the proposed scheme in improving offloading utility and reliability.

Acknowledgements. This work was supported in part by the National Natural Science Foundation of China under Grant No. 62172224, in part by the Natural Science Foundation of Jiangsu Province under Grant No. BK20220138, in part by the China Postdoctoral Science Foundation under Grant Nos. BX2021128, 2021T140327, 2020M680068, in part by the Fundamental Research Funds for the Central Universities under Grant Nos. 30922010318 and 30922010406, in part by the Postdoctoral Science Foundation of Jiangsu Province under Grant No. 2021K066A, in part by the Open Research Fund of the State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences under Grant No. CAR-CHA202105, in part by the Future Network Scientific Research Fund Project under Grant No. FNSRFP-2021-YB-6, and in part by the Open Research Fund of Engineering Research Center of Software/Hardware Co-Design Technology and Application, Ministry of Education (East China Normal University) under Grant No. OP202203.

References

1. Bhat, G., Bagewadi, K., Lee, H.G., Ogras, U.Y.: REAP: runtime energy-accuracy optimization for energy harvesting IoT devices. In: ACM/IEEE DAC, pp. 1–6 (2019)
2. Taleb, T., Samdanis, K., Mada, B., Flinck, H., Dutta, S., Sabella, D.: On multi-access edge computing: a survey of the emerging 5G network edge cloud architecture and orchestration. In: IEEE CMOST, pp. 1657–1681 (2017)
3. Kuang, Z., Li, L., Gao, J., Zhao, L., Liu, A.: Partial offloading scheduling and power allocation for mobile edge computing systems. In: IEEE IoT, pp. 6774–6785 (2019)
4. Wang, J., Hu, J., Min, G., Zomaya, A.Y., Georgalas, N.: Fast adaptive task offloading in edge computing based on meta reinforcement learning. In: IEEE TPDS, pp. 242–253 (2021)

5. Chang, Z., Liu, L., Guo, X., Sheng, Q.: Dynamic resource allocation and computation offloading for IoT fog computing system. In: IEEE TII, pp. 3348–3357 (2021)
6. Gupta, S., Chakareski, J.: Lifetime maximization in mobile edge computing networks. In: IEEE TVT, pp. 3310–3321 (2020)
7. Tran, T.X., Pompili, D.: Joint task offloading and resource allocation for multi-server mobile-edge computing networks. In: IEEE TVT, pp. 856–868 (2019)
8. Salehi, M.: DRVS: power-efficient reliability management through dynamic redundancy and voltage scaling under variations. In: IEEE ISLPED, pp. 225–230 (2015)
9. Choi, Y., Park, S., Bahk, S.: Multi channel random access in OFDMA wireless networks. In: IEEE JSAC, pp. 603–613 (2006)
10. Liao, Z., Peng, J., Huang, J., Wang, J., Sharma, P.K., Ghosh, U.: Distributed probabilistic offloading in edge computing for 6G-Enabled massive internet of things. In: IEEE IoT, pp. 5298–5308 (2021)
11. Li, L., Cong, P., Cao, K., Zhou, J.: Feedback control of real-time EtherCAT networks for reliability enhancement in CPS. In: ACM/IEEE DATE, pp. 688–693 (2018)
12. Zhou, J., Hu, X.S., Ma, Y., Sun, J., Wei, T., Hu, S.: Improving availability of multicore real-time systems suffering both permanent and transient faults. IEEE TC **68**(12), 1785–1801 (2019)
13. Zhou, J., Cao, K., Zhou, X., Chen, M., Wei, T., Hu, S.: Throughput-conscious energy allocation and reliability-aware task assignment for renewable powered in-situ server systems. IEEE TCAD **41**(3), 516–529 (2022)
14. Zhao, Z.: A novel framework of three-hierarchical offloading optimization for MEC in industrial IoT networks. In: IEEE TII, pp. 5424–5434 (2020)
15. Zhang, Y., Zhou, J., Sun, L., Mao, J., Sun, J.: A novel firefly algorithm for scheduling bag-of-tasks applications under budget constraints on hybrid clouds. IEEE Access **7**, 151888–151901 (2019)



EdgeViT: Efficient Visual Modeling for Edge Computing

Zekai Chen¹, Fangtian Zhong², Qi Luo¹, Xiao Zhang¹, and Yanwei Zheng¹(✉)

¹ School of Computer Science and Technology, Shandong University, Qingdao, China
{zjf5176, xiaozhang, zhengyw}@psu.edu

² Pennsylvania State University, State College, PA, USA

Abstract. With the rapid growth of edge intelligence, a higher level of deep neural network computing efficiency is required. Visual intelligence, as the core component of artificial intelligence, is particularly worth more exploration. As the cornerstone of modern visual modeling, convolutional neural networks (CNNs) have greatly developed in the past decades. Variants of light-weight CNNs have also been proposed to address the challenge of heavy computing in mobile settings. Though CNNs' spatial inductive biases allow them to learn representations with fewer parameters across different vision tasks, these models are spatially local. To acquire a next-level model performance, vision transformer (ViT) is now a viable alternative due to the potential of multi-head attention mechanism. In this work, we introduce EdgeViT, an accelerated deep visual modeling method that incorporates the benefits of CNNs and ViTs in a light-weight and edge-friendly manner. Our proposed method can achieve top-1 accuracy of 77.8% using only 2.3 million parameters, 79.2% using 5.6 million parameters on ImageNet-1k dataset. It can achieve mIoU up to 78.3 on PASCAL VOC segmentation while only using 3.1 million parameters which is only half of MobileViT parameter budget.

Keywords: Edge computing · Vision transformer · Lite computation

1 Introduction

Edge computing is gaining traction recently as new use cases emerge, notably with the introduction of 5G [43]. Specifically, edge computing combined with artificial intelligence [37, 56] enables faster computation and insights, improved data protection, and effective control over continuous operations. Automated optical inspection, for example, plays a critical role in manufacturing lines. With the help of an automated Edge visual model, it is possible to detect faulty portions of assembled components on a production line. Without relying on massive volumes of cloud-based data transmission, an edge based automated optical inspection enables for both accurate and ultrafast pipeline processing. Besides the manufacturing industry, many other real-world applications (e.g., autonomous vehicles,

facial recognition, emergency medical care, etc.) also require such visual recognition tasks (e.g., object detection and semantic segmentation) to run on edge devices in a timely fashion.

As the cornerstone of modern deep visual representation learning, convolutional neural networks (CNNs) have gained massive success in the past decades. Recent methods such as vision transformers (ViTs, [14, 50]) have also been proposed with the development of self-attention mechanism [50]. It has been extremely successful in NLP since huge transformer language models [1, 13] have substantially revolutionized the discipline. Likewise, vision transformers (ViTs) [14], are now a viable alternative to convolutional neural networks (CNNs) to learn visual representations and have been the *defacto* benchmark backbones for many downstream vision tasks. Generally, ViT divides an image into a series of non-overlapping patches and then applies multi-head self-attention transformers to learn inter-patch representations. According to previous research, the performance gains of these ViT networks are usually at the expense of model size (#Parameters) and computation complexity (#MACs). However, in most edge infrastructures, *the computation and memory capacity is restricted by the edge or IoT devices' capability*. As a result, large complex deep neural networks have to be compressed (e.g., pruning [17, 22, 32], model quantization [27, 51], etc.) prior to the deployment to the Edge hardware, which almost inevitably results in model performance degradation. To be effective, visual models in such infrastructure should be **light** and **fast** at inference.

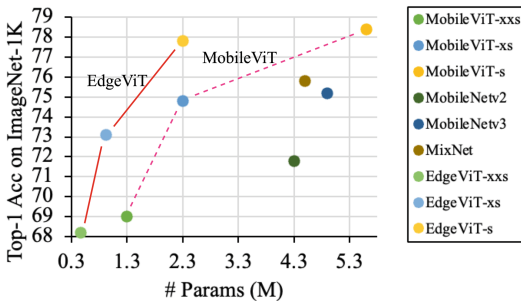


Fig. 1. EdgeViT produces higher Top-1 accuracy on ImageNet-1K than other light-weight CNN models and also MobileViT with significantly smaller memory footprint.

Recent light-weight design of CNNs and ViTs have been proposed and demonstrated to lift performance in many mobile settings [4, 25, 38, 42]. Compared to CNNs, ViTs are more heavy-weight which are often harder to optimize [55], needs extensive data augmentation and L2 regularization to prevent overfitting [48, 52]. For example, a ViT-based segmentation network [40] learns about **345M** parameters and achieves similar performance as the CNN-based network, DeepLabv3 [3], with **59M** parameters. The lack of image-specific inductive bias [55] (attention vs. convolutions) resulting in an appetite of *parameters* also limits the capabilities of ViTs to be applied in edge computing scenarios compared to CNNs. Therefore, hybrid approaches that combine the benefits of convolutions (e.g., spatial inductive bias and less sensitive to data augmentation) and transformers (e.g., input-adaptive weighting and global relationship modeling) are gaining interest [4, 12, 38, 55], among which, MobileViT [38] is one

Recent light-weight design of CNNs and ViTs have been proposed and demonstrated to lift performance in many mobile settings [4, 25, 38, 42]. Compared to CNNs, ViTs are more heavy-weight which are often harder to optimize [55], needs extensive data augmentation and L2 regularization to prevent overfitting [48, 52]. For example, a ViT-based segmentation network [40] learns about **345M** parameters and achieves similar performance as the CNN-

representative that can produce competitive model performance while maintain a low memory footprint. Specifically, the MobileViT design (see Fig. 2 right panel) follows a local-global [54] information **sequential** modeling paradigm. Standard convolution involves local representation learning initially, and then transformers learn global representations of the images. Nevertheless, edge devices require even more stringent memory footprint and computation complexity (e.g., some sensor devices only support <1G FLOPs and a latency lower than 15ms [43]) to keep its efficacy. Inspired by MobileViT [38], we present **EdgeViT**, a new architecture that separates the local-global representation learning into two parallel branches rather than serializing through the embedding dimensionality. It shrinks the feature map size to further reduce the model size and total computation amount (#MACs) while not severely hindering the model performance. The increased degree of parallelism also advances inference speed and lowers the latency.

Our extensive experiments show that (also see Fig. 1), our proposed EdgeViT can achieve a top-1 accuracy of **77.8%** (1% less than MobileViT’s accuracy) on ImageNet-1K dataset (image classification) [41], while only uses **46.56%** of the parameter budget and **62.41%** of the computing budge (#MACs) of MobileViT under the same training recipe. Integrated with a DeepLabV3 decoded head, our method can achieve a **78.9%** mIOU on PASCAL VOC 2012 dataset [15], which is **4.2%** better than MobileNetV2 [42].

2 Related Work

2.1 Vision Transformers

Transformers [14, 50] for large-scale image recognition and shown that ViTs may reach CNN-level accuracy without image-specific inductive bias on extremely large-scale datasets (e.g., JFT-300M). ViTs [14] can be trained on the ImageNet dataset to achieve CNN-level performance [48, 49, 60] with significant data augmentation, intensive L2 regularization, and distillation. However, unlike CNNs, ViTs have poor optimizability and are difficult to train. [2, 11, 16, 31, 52, 57] reveal that the lack of spatial inductive biases in ViTs is the cause of the poor optimizability. The use of convolutions in ViTs to include such biases increases their stability and performance. To take advantage of convolutions and transformers, various designs have been investigated. ViT-C [55], for example, adds an early convolutional stem to help ViT see better. CvT [53] alters multi-head attention in transformers by replacing linear projections with depth-wise separable convolutions. BoTNet [44] replaces ResNet’s bottleneck unit’s normal 33% convolution with multi-head attention. ConViT [12] uses gated positional self-attention to add soft convolutional inductive biases. PiT [23] is a depth-wise convolution-based pooling layer that extends ViT. Despite the fact that with considerable augmentation, these models may compete with CNNs, the majority of them are heavy-weight. On the ImageNet-1k dataset, PiT and CvT, for example, learn 6.1 and 1.7× more parameters than EfficientNet [46] and achieve similar performance (top-1 accuracy of roughly 81.6%). Furthermore, when these models

are scaled down to create light-weight ViT models, their performance lags well below that of light-weight CNNs. ImageNet-1k accuracy of PiT is 2.2% worse than MobileNetv3 [24] with a parameter budget of roughly 6 million. Most recent MobileViT [38] incorporates MobileNetv2 blocks with ViTs which has a competitive generalization capability while also light-weight.

2.2 Light-Weight CNNs

Standard convolutional layers serve as the foundation of CNNs. Numerous factorization based techniques have been suggested to reduce the computational cost of this layer and make it mobile-friendly [8, 28, 39]. Modern light-weight CNNs for mobile vision tasks, such as MobileNets [24, 25, 42], ShuffleNetv2 [36], MixNet [47], and MNASNet [45], frequently use separable convolutions of [8, 46]. These networks can quickly substitute the heavy-weight backbones in current task-specific models (such DeepLabv3 [3]) to reduce network size and improve latency, such as ResNet [20]. Despite these advantages, these approaches have a significant disadvantage in that they are spatially confined. In this work, we aim to combine the advantages of both convolutions and transformers (e.g., versatile and simple training) to construct a deep visual model that works under critical computational latency requirements such as edge.

2.3 Efficient Computing in Deep Learning

Another strategy to accomplish efficient inference is to compress and speed the current huge models, in addition to directly building efficient models [32]. To speed up model inference, some have suggested pruning individual neurons [17, 18], entire channels [21, 22, 33], or the network as a whole [9, 51]. Recently, model acceleration and compression have also been automated using AutoML [21, 32, 51]. Multi-tasking is also a viable alternative [6, 7]. These research are orthogonal to our work because they all condense already-built models. Instead of compressing or accelerating an existing model, we are more interested in investigating how to harness the domain knowledge to develop an effective architecture from the initial.

3 EdgeViT: Efficient Visual Modeling for Edge

MobileViT [38] adopts a local-global representation learning paradigm to incorporate the benefits of CNNs and ViTs, however, the sequential modeling design may not be the optimal choice. As shown in Fig. 2 (middle), our EdgeViT block follows a double-branch design. It splits the original input feature maps $\mathbf{X} \in \mathbb{R}^{c \times h \times w}$ into two portions along the channel dimension c , followed by two attention branches: one convolution branch for extracting information in a restricted neighborhood and one typical transformer attention branch for capturing global-wise dependencies. Following ViT [14], the feature maps are divided into regular non-overlapping patches, which are often considered as the basic

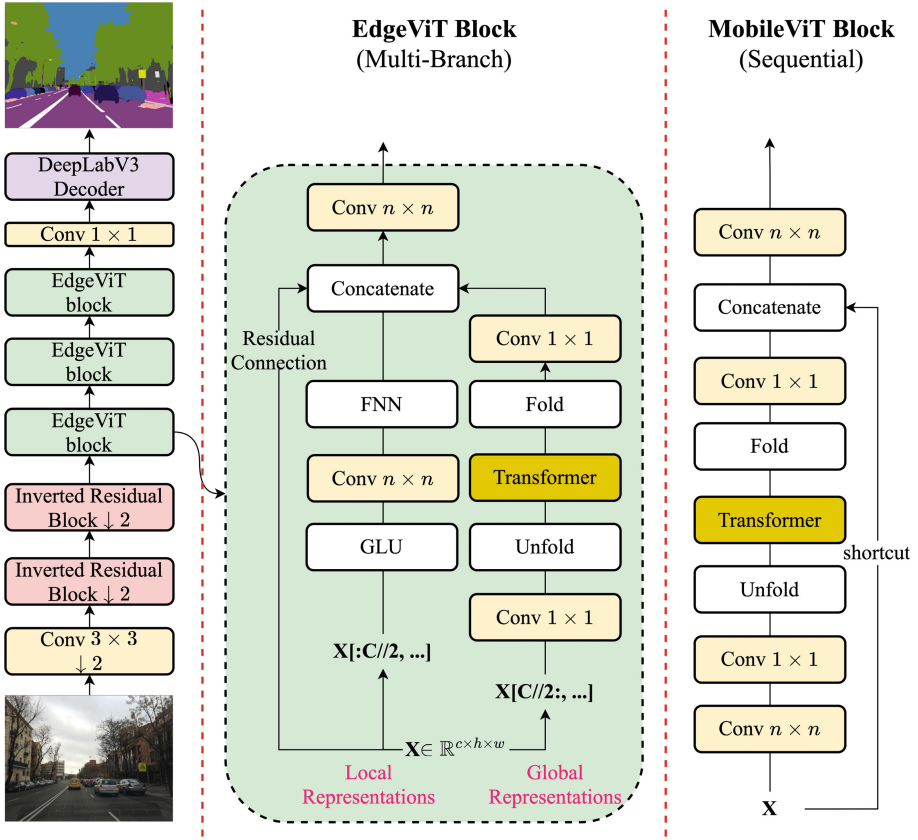


Fig. 2. EdgeViT. An illustration of how the architecture design of EdgeViT compares to MobileViT. For EdgeViT, we adopt a multi-branch design for the attention computation by splitting the input $\mathbf{X} \in \mathbb{R}^{c \times h \times w}$ across the channel dimension. The convolutional operations are used for neighboring patches representation modeling and the transformer is used for global representation modeling. As such, nearly half computation complexity of the transformer portion is simply reduced.

processing units of Transformers. Convolution over the feature maps is one obvious solution for the local-representation modeling. The diagonal groups can be simply covered by the module with a sliding window. Some light-weight and accelerated implementation of CNNs are utilized in this case.

3.1 Self-attention in Vision Transformers

The vanilla multi-head self-attention mechanism was originally proposed by [50]. For a sequence of token representations $\mathbf{X} \in \mathbb{R}^{n \times d}$ (with sequence length n and dimensionality d), the self-attention function firstly projects them into queries $\mathbf{Q} \in \mathbb{R}^{n \times d_k}$, keys $\mathbf{K} \in \mathbb{R}^{n \times d_k}$ and values $\mathbf{V} \in \mathbb{R}^{n \times d_v}$, h times with different,

learned linear projections to d_k , d_k and d_v dimensions, respectively. Then a particular scaled dot-product attention was computed to obtain the weights on the values as:

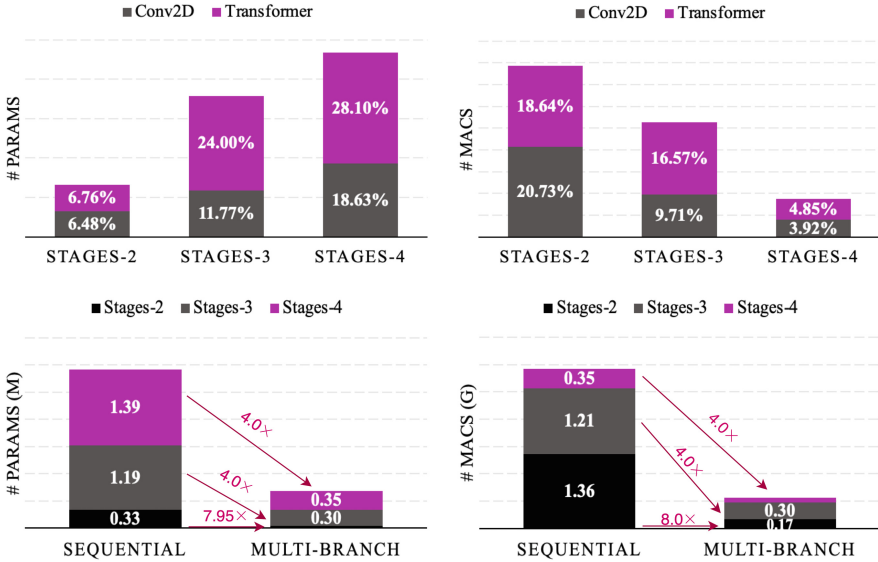


Fig. 3. As Fig. 2 shows, a typical EdgeViT is composed of 5 stages (from 0 to 4). Using an example of image with size (3, 512, 512), the top panel shows that in all EdgeViT blocks, transformer occupies a large portion of both network parameters and computations. The bottom panel demonstrates that multi-branch design significantly optimizes the model size and reduces the overall compute cost.

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V} \tag{1}$$

Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. With a concatenated computing way, the final output of multi-head attention is as following:

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O \tag{2}$$

in which, h is the number of total heads. Each head is defined as:

$$\text{head}_i = \text{Attention}(\mathbf{Q}W_i^Q, \mathbf{K}W_i^K, \mathbf{V}W_i^V) \tag{3}$$

where the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d \times d_k}$, $W_i^K \in \mathbb{R}^{d \times d_k}$, $W_i^V \in \mathbb{R}^{d \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d}$.

3.2 Complexity Analysis

LSRA [5, 54] has demonstrated the effectiveness of multi-branch attention in capturing global and local context patterns, especially under mobile computational constraints. Figure 3 compares the amount of parameters and computational complexity between MobileViT block and our EdgeViT block across multiple stages (see the main architecture of EdgeViT in Fig. 2 (left)). Across all stages, the transformer occupies over 59% of the total model memory footprint and about 40% of the total GFLOPs, and this trend will become more significant as the depth of transformer layers in each stage increases.

Table 1. Comparison with both light-weight and heavy-weight CNNs and also MobileViT on image classification (validation set).

	Model	#Params (M)	IN1k-Top1 (%)
Light weight	MobileNetv1	2.6	68.4
	MobileNetv2	2.6	69.8
	MobileNetv3	2.5	67.4
	ShuffleNetv2	2.3	69.4
	ESPNetv2	2.3	69.2
	MobileViT-xs	2.3	74.8
	EdgeViT-xs (ours)	0.92	<u>73.1</u>
Heavy weight	DenseNet-169	14	76.2
	EfficientNet-B0	5.3	76.3
	ResNet-101	44.5	77.4
	ResNet-101-SE	49.3	77.6
	MobileViT-s	5.6	78.4
	EdgeViT-s (ours)	2.3	<u>77.8</u>

The bottom panel in Fig. 3 further demonstrates that a multi-branch based architecture will compress the entire attention procedure over $4\times$ for later stages and over $8\times$ for early stages across both model size and computational complexity.

4 Experimental Results

In this section, we analyze EdgeViT’s performance on the ImageNet-1k dataset and also a visual segmentation task, demonstrating that it is competitive to other state-of-the-art methods. We demonstrate EdgeViT’ general applicability and edge device-friendly in following sections.

4.1 Image Classification on ImageNet-1K Dataset

Implementation Settings. On the ImageNet-1k classification dataset [41], we train EdgeViT from scratch. The training and validation sets of IN-1K

provide 1.28 million and 50 thousand images, respectively. With an effective batch size of 1,024 images, the MobileViT networks are trained using PyTorch for 300 epochs on 8 NVIDIA A10 GPUs using the AdamW optimizer [35], label smoothing cross-entropy loss (smoothing=0.1), and multi-scale sampler ($\mathcal{S} = \{(160, 160), (192, 192), (256, 256), (288, 288), (320, 320)\}$). For the first 3000 iterations, the learning rate is increased from 0.0002 to 0.002, and then it is annealed to 0.0002 using a cosine schedule [34]. We employ a L2 weight decay equals to 0.01. The performance is assessed using a single crop top-1 accuracy and baseline data augmentation (i.e., random scaled cropping and horizontal flipping). An exponential moving average of the model weights is employed for inference following [38].

Table 2. Comparison with ViTs based models on ImageNet-1k validation set. (aug) stands for more advanced augmentation methods combined with baseline augmentation methods (e.g. MixUp [58], RandAug [10], and CutMix [59]).

Model	#Params(M)	IN1k-Top1(%)
DeiT	5.7	68.7
	5.7 (aug)	72.2
	10.0 (aug)	75.9
T2T	4.3 (aug)	71.7
	6.9 (aug)	76.5
PiT	10.6	72.4
	4.9 (aug)	73.0
	10.6 (aug)	78.1
Mobile-former	4.6 (aug)	72.8
	9.4 (aug)	76.7
Cross-ViT	6.9 (aug)	73.4
	8.5 (aug)	73.8
CeiT	6.4 (aug)	76.4
ViL	6.7 (aug)	76.7
LocalViT	7.7 (aug)	76.1
PVT	13.2 (aug)	75.1
ConViT	10.0 (aug)	76.7
BoTNet	20.8	77.0
BoTNet	20.8 (aug)	78.3
MobileViT	2.3 (xs)	74.8
	5.6 (s)	78.4
EdgeViT	0.92 (xs)	73.1
	2.3 (s)	77.8
	5.6 (xl)	79.2

Table 3. Comparison with other SOTAs on segmentation w/ DeepLabv3.

Model	#Params (M)	mIOU
MobileNetv1	11.2	75.3
MobileNetv2	4.5	75.7
MobileViT-xs	1.9	73.6
MobileViT-xs	2.9	77.1
EdgeViT-xs	1.3	76.5
ResNet-101	58.23	80.5
MobileViT-s	6.4	79.1
EdgeViT-s	3.1	78.3

Comparison with CNNs. In different model memory footprints (MobileNetv1 [25], MobileNetv2 [42], ShuffleNetv2 [36], ESPNetv2 [39], MobileNetv3 [24] and MobileViT [38], EdgeViT outperforms all light-weight CNNs, and reaches a competitive performance compared to MobileViT, however, only using a over $2\times$ smaller network size as shown in Table 1. Specifically, EdgeViT outperforms MobileNetv2 by 4.73%, ShuffleNetv2 by 5.33%, and MobileNetv3 by 8.46% on the ImageNet-1K validation set for a model size of less than 1 million parameters (0.92M). The results also demonstrate that EdgeViT outperforms heavy-weight CNNs like ResNet [20], DenseNet [26], ResNet-SE, and EfficientNet [46]. For a similar set of criteria, EdgeViT is, for instance, 1.97% more accurate than EfficientNet.

Comparison with ViTs. We compare our EdgeViT with many ViT variants including DeiT [48], T2T [57], PVT [52], CAiT [49], CrossViT [2], LocalViT [29], PiT [23]. EdgeViT performs better with fewer parameters and basic augmentation than other ViT variants that gain a lot from advanced augmentation (e.g., PiT with basic augmentation vs. advanced: 72.4 vs. 78.1 (see Table 2). For example, EdgeViT outperforms DeiT by 13.25% while being $2.5\times$ smaller. All these evidences show that EdgeViTs are just like CNNs, are reliable and easy to optimize. Again, it is easy to adapt them to various other tasks and datasets.

4.2 Semantic Segmentation on PASCAL VOC Challenge

Implementation Settings. We adopt DeepLabv3 [3] as the decode head and EdgeViT as the backbone to train from scratch for semantic segmentation. On the PASCAL VOC 2012 dataset [15], we fine-tune EdgeViT using AdamW with cross-entropy loss. We also use additional annotations and data from [19, 30], respectively, in accordance with a common training approach [3, 38]. Mean intersection over union is used to evaluate the performance on the validation set (mIOU). Same to training on ImageNet, we train all models on NVIDIA A10 GPUs with Pytorch.

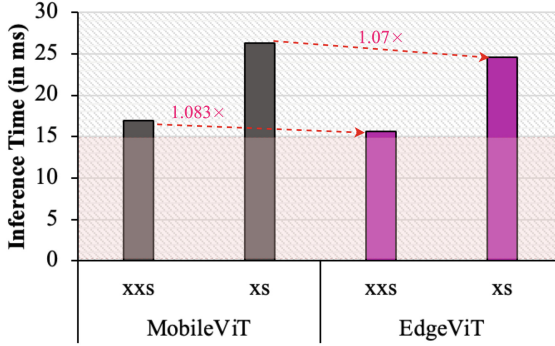


Fig. 4. Inference time of EdgeViT compared to MobileViT on segmentation tasks. EdgeViT is very close to the very restrictive many edge latency requirements (shaded area: inference time < 15ms). Tests are based on AMD EPYC 7R13 Processor, using 1 core with 2 threads and CPU MHz@2405.544.

Comparison with MobileViTs. Table 3 demonstrates how DeepLabv3 incorporated with EdgeViT is more efficient and superior than other SOTAs. When EdgeViT-xs is utilized as a backbone instead of MobileNetv2, DeepLabv3 performs 1.0% better and has a $3.46\times$ size reduction. Additionally, EdgeViT models with ResNet-101 with comparable performance while requiring $18.78\times$ fewer parameters, indicating EdgeViT is a strong but super light-weight backbone. Additionally, the results in Table 3 demonstrate that EdgeViT can achieve a competitive performance as MobileViT while only uses half of the parameters budget. It is also important to notice that the real-time inference time of EdgeViT is only slightly faster (see Fig. 4). than MobileViT. The main causes of this difference are two parts. First, particular CPU cores or CUDA kernels are accessible and used right out of the box in ViTs to improve the scalability and efficiency of transformers on CPUs or GPUs. Additionally, one of the device-level improvements that CNNs benefit from is batch normalization fusion with convolutional layers. These modifications enhance memory access and latency. However, there are not any such specialized and effective procedures for transformers right now.

5 Discussion

In this work, we propose EdgeViT, a light-weight deep network for visual modeling in edge computing. The core is based on multi-branch attention architecture which significantly reduces the model size and accelerates the computation. This method can achieve competitive performance on wide range of downstream tasks while maintaining a very low memory footprint which is naturally edge-friendly. Nevertheless, a further combination with deep compressing or quantization or distillation techniques are worth investigation.

References

1. Brown, T.B., et al.: Language models are few-shot learners. In: *NeurIPS (2020)*
2. Chen, C.F., Fan, Q., Panda, R.: CrossViT: cross-attention multi-scale vision transformer for image classification. In: *ICCV*, pp. 347–356 (2021)
3. Chen, L.C., Papandreou, G., Schroff, F., Adam, H.: Rethinking atrous convolution for semantic image segmentation. *ArXiv abs/1706.05587* (2017)
4. Chen, Y., Dai, X., Chen, D., Liu, M., Dong, X., Yuan, L., Liu, Z.: Mobile-former: bridging mobilenet and transformer. *arXiv abs/2108.05895* (2021)
5. Chen, Z., Chen, D., Yuan, Z., Cheng, X., Zhang, X.: Learning graph structures with transformer for multivariate time-series anomaly detection in IoT. *IEEE Internet Things J.* **9**, 9179–9189 (2022)
6. Chen, Z., Ji, E., Zhang, X., Sheng, H., Cheng, X.: Multi-task time series forecasting with shared attention. In: *ICDMW*, pp. 917–925 (2020)
7. Chen, Z., Shi, M., Zhang, X., Ying, H.: Asm2tv: an adaptive semi-supervised multi-task multi-view learning framework. In: *AAAI* (2022)
8. Chollet, F.: Xception: deep learning with depthwise separable convolutions. In: *CVPR*, pp. 1800–1807 (2017)
9. Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R., Bengio, Y.: Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1. *arxiv abs/1602.02830* (2016)
10. Cubuk, E.D., Zoph, B., Mané, D., Vasudevan, V., Le, Q.V.: Autoaugment: learning augmentation strategies from data. In: *CVPR*, pp. 113–123 (2019)
11. Dai, Z., Liu, H., Le, Q.V., Tan, M.: CoAtNet: marrying convolution and attention for all data sizes. In: *NeurIPS* (2021)
12. d’Ascoli, S., Touvron, H., Leavitt, M.L., Morcos, A.S., Biroli, G., Sagun, L.: Convit: improving vision transformers with soft convolutional inductive biases. In: *ICML* (2021)
13. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: *NAACL* (2019)
14. Dosovitskiy, A., et al.: An image is worth 16x16 words: transformers for image recognition at scale. In: *ICLR* (2020)
15. Everingham, M., Eslami, S.M.A., Gool, L.V., Williams, C.K.I., Winn, J.M., Zisserman, A.: The pascal visual object classes challenge: a retrospective. *IJCV* **111**, 98–136 (2014)
16. Graham, B., et al.: LeViT: a vision transformer in convnet’s clothing for faster inference. In: *ICCV*, pp. 12239–12249 (2021)
17. Han, S., Mao, H., Dally, W.J.: Deep compression: compressing deep neural network with pruning, trained quantization and huffman coding. In: *NeurIPS* (2016)
18. Han, S., Pool, J., Tran, J., Dally, W.J.: Learning both weights and connections for efficient neural network. In: *NeurIPS* (2015)
19. Hariharan, B., Arbeláez, P., Bourdev, L.D., Maji, S., Malik, J.: Semantic contours from inverse detectors. In: *ICCV*, pp. 991–998 (2011)
20. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: *CVPR*, pp. 770–778 (2016)
21. He, Y., Lin, J., Liu, Z., Wang, H., Li, L.-J., Han, S.: AMC: AutoML for model compression and acceleration on mobile devices. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) *ECCV 2018*. LNCS, vol. 11211, pp. 815–832. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01234-2_48

22. He, Y., Zhang, X., Sun, J.: Channel pruning for accelerating very deep neural networks. In: ICCV, pp. 1398–1406 (2017)
23. Heo, B., Yun, S., Han, D., Chun, S., Choe, J., Oh, S.J.: Rethinking spatial dimensions of vision transformers. In: ICCV, pp. 11916–11925 (2021)
24. Howard, A.G., et al.: Searching for mobilenetv3. In: ICCV, pp. 1314–1324 (2019)
25. Howard, A.G., et al.: Mobilenets: efficient convolutional neural networks for mobile vision applications. arXiv abs/1704.04861 (2017)
26. Huang, G., Liu, Z., Weinberger, K.Q.: Densely connected convolutional networks. In: CVPR, pp. 2261–2269 (2017)
27. Jacob, B., et al.: Quantization and training of neural networks for efficient integer-arithmetic-only inference. In: CVPR, pp. 2704–2713 (2018)
28. Jin, J., Dundar, A., Culurciello, E.: Flattened convolutional neural networks for feedforward acceleration. CoRR abs/1412.5474 (2015)
29. Li, Y., Zhang, K., Cao, J., Timofte, R., Gool, L.V.: LocalViT: bringing locality to vision transformers. arXiv abs/2104.05707 (2021)
30. Lin, T.-Y., et al.: Microsoft COCO: common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8693, pp. 740–755. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10602-1_48
31. Liu, Z., et al.: Swin transformer: hierarchical vision transformer using shifted windows. In: ICCV, pp. 9992–10002 (2021)
32. Liu, Z., Mu, H., Zhang, X., Guo, Z., Yang, X., Cheng, K., Sun, J.: MetaPruning: meta learning for automatic neural network channel pruning. In: ICCV, pp. 3295–3304 (2019)
33. Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., Zhang, C.: Learning efficient convolutional networks through network slimming. In: ICCV, pp. 2755–2763 (2017)
34. Loshchilov, I., Hutter, F.: SGDR: stochastic gradient descent with warm restarts. In: ICLR (2016)
35. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: ICLR (2019)
36. Ma, N., Zhang, X., Zheng, H.-T., Sun, J.: ShuffleNet V2: practical guidelines for efficient CNN architecture design. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) Computer Vision – ECCV 2018. LNCS, vol. 11218, pp. 122–138. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01264-9_8
37. Mao, Y., You, C., Zhang, J., Huang, K., Letaief, K.B.: A survey on mobile edge computing: the communication perspective. IEEE CST **19**, 2322–2358 (2017)
38. Mehta, S., Rastegari, M.: MobileViT: light-weight, general-purpose, and mobile-friendly vision transformer. arXiv abs/2110.02178 (2021)
39. Mehta, S., Rastegari, M., Caspi, A., Shapiro, L., Hajishirzi, H.: ESPNet: efficient spatial pyramid of dilated convolutions for semantic segmentation. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11214, pp. 561–580. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01249-6_34
40. Ranftl, R., Bochkovskiy, A., Koltun, V.: Vision transformers for dense prediction. In: ICCV, pp. 12159–12168 (2021)
41. Russakovsky, O., et al.: ImageNet large scale visual recognition challenge. IJCV **115**, 211–252 (2015)
42. Sandler, M., Howard, A.G., Zhu, M., Zhmoginov, A., Chen, L.C.: MobileNetv 2: inverted residuals and linear bottlenecks. In: CVPR, pp. 4510–4520 (2018)
43. Shi, W., Cao, J., Zhang, Q., Li, Y., Xu, L.: Edge computing: Vvision and challenges. IEEE IoTJ **3**, 637–646 (2016)
44. Srinivas, A., Lin, T.Y., Parmar, N., Shlens, J., Abbeel, P., Vaswani, A.: Bottleneck transformers for visual recognition. In: CVPR, pp. 16514–16524 (2021)

45. Tan, M., Chen, B., Pang, R., Vasudevan, V., Le, Q.V.: MnasNet: platform-aware neural architecture search for mobile. In: CVPR, pp. 2815–2823 (2019)
46. Tan, M., Le, Q.V.: EfficientNet: rethinking model scaling for convolutional neural networks. In: ICML (2019)
47. Tan, M., Le, Q.V.: MixConv: mixed depthwise convolutional kernels. arXiv abs/1907.09595 (2019)
48. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., J'egou, H.: Training data-efficient image transformers & distillation through attention. In: ICML (2021)
49. Touvron, H., Cord, M., Sablayrolles, A., Synnaeve, G., J'egou, H.: Going deeper with image transformers. In: ICCV, pp. 32–42 (2021)
50. Vaswani, A., et al.: Attention is all you need. In: NeurIPS (2017)
51. Wang, K., Liu, Z., Lin, Y., Lin, J., Han, S.: HAQ: hardware-aware automated quantization with mixed precision. In: CVPR, pp. 8604–8612 (2019)
52. Wang, W., et al.: Pyramid vision transformer: a versatile backbone for dense prediction without convolutions. In: ICCV, pp. 548–558 (2021)
53. Wu, H., Xiao, B., Codella, N.C.F., Liu, M., Dai, X., Yuan, L., Zhang, L.: CvT: introducing convolutions to vision transformers. In: ICCV, pp. 22–31 (2021)
54. Wu, Z., Liu, Z., Lin, J., Lin, Y., Han, S.: Lite transformer with long-short range attention. In: ICLR (2020)
55. Xiao, T., Singh, M., Mintun, E., Darrell, T., Dollár, P., Girshick, R.B.: Early convolutions help transformers see better. In: NeurIPS (2021)
56. Yu, H., et al.: FedHAR: semi-supervised online learning for personalized federated human activity recognition. *IEEE Transactions on Mobile Computing* (2021)
57. Yuan, L., et al.: Tokens-to-token ViT: training vision transformers from scratch on imagenet. In: ICCV, pp. 538–547 (2021)
58. Zhang, H., Cissé, M., Dauphin, Y., Lopez-Paz, D.: mixup: beyond empirical risk minimization. arXiv abs/1710.09412 (2018)
59. Zhong, Z., Zheng, L., Kang, G., Li, S., Yang, Y.: Random erasing data augmentation. In: AAAI (2020)
60. Zhou, D., et al.: DeepViT: towards deeper vision transformer. arXiv abs/2103.11886 (2021)



Joint Optimization of Computation Task Allocation and Mobile Charging Scheduling in Parked-Vehicle-Assisted Edge Computing Networks

Wenqiu Zhang^{1,2(✉)}, Ran Wang^{1,2}, Changyan Yi^{1,2}, and Kun Zhu^{1,2}

¹ Nanjing University of Aeronautics and Astronautics, Nanjing, China
{fzwq1998, wangran, changyan.yi, zhukung}@nuaa.edu.cn

² Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing, China

Abstract. In this paper, we study the joint optimization of task allocation and charging scheduling of mobile charging vehicles (MCVs) for parked-vehicle-assisted edge computing networks. In the proposed model, a group of electric vehicles (EVs) that have been parked for a long time must be recharged to their expected energy level within a specified time frame. Meanwhile, an optimal set of parked vehicles (PVs) is selected to compute a machine learning task utilizing their hardware resources and local data while satisfying the task's training performance requirements. Within the calculated time window, an MCV is dispatched to provide power replenishment to the PVs. By jointly deciding the task allocation and MCV charging sequence, the proposed model seeks to minimize the total energy consumption of the parked vehicular network, which includes the PV computation and MCV traveling consumption, subject to the PVs' expected energy level, task target utility and time window. To address this joint optimization problem, a marginal-product-based algorithm is designed, where a deep reinforcement learning method is integrated to solve the MCV scheduling problem. Simulation results demonstrate that the proposed method can efficiently solve the problem and outperform the compared algorithms in terms of energy consumption.

Keywords: Parked vehicular network · Edge computing · Mobile charging · Deep reinforcement learning

1 Introduction

Electric vehicles (EVs) have recently received considerable attention due to their environmental friendliness and low carbon emissions compared to conventional

This work is supported by the National Natural Science Foundation of China under grant No. 62171218.

diesel locomotives. Therefore, a significant increase in the adoption of EVs has been observed in recent years. Meanwhile, with the upgrading of vehicle hardware and the development of artificial intelligence technology, vehicular edge training has become a future development trend of machine learning, where parked EVs can utilize idle computing resources to undertake learning-related tasks (e.g., optimizing parameters of an autopilot model by computing local video data).

However, one of the key barriers limiting the expansion of the EV market is the issue of power replenishment. Furthermore, in addition to the power consumed by vehicle movement, long-term high-power machine learning training incurs additional charging pressure. The mainstream solution for this problem is fixed charging stations (FCSs), where users must drive to a suitable charging station and wait for the charging process to be completed. On the one hand, the construction and maintenance of FCSs require expensive capital investment. On the other hand, due to the lack of flexibility, FCSs cannot fully satisfy users' charging demands, which vary over time and space.

Since previously methods cannot fundamentally address the EV charging problem, recent advances in charging technology have inspired the emergence of mobile charging services (MCSs) as a promising alternative solution. Mobile charging vehicles (MCVs), which have energy storage, can provide charging services for EV customers. The mobility of MCVs enables instant adaptation to the charging needs of different regions; furthermore, the low price of MCVs compared to FCSs makes them even more competitive. Issues related to MCVs have been discussed in recent research [1, 2]. Traditional optimization methods (e.g., queuing theory) and learning-based algorithms are employed to solve the modeled problems to maximize the charging benefits while minimizing the cost.

However, the existing researches considers only typical charging scenarios; the emergence of edge training introduces a new charging scenario. In [3], the concept and application of vehicular edge training are explored in detail. Heterogeneous EVs that undertake training tasks generate additional charging demands, and their state of charge varies over time. Moreover, dispatched MCVs incur additional power consumption and costs. Therefore, the optimal set of EVs must be selected to collaboratively and continuously execute training tasks while guaranteeing that the performance requirements of the machine learning model are satisfied. Moreover, another key issue of MCVs is to determine the charge sequence and the amount of charging to provide when serving EVs.

Furthermore, EVs must perform high-power training tasks for long periods and feed the trained parameters of the learning model back to the central control platform. Therefore, any unpredictable EV failure may cause failed training results. Hence, to guarantee that all selected EVs work continuously during the task period, the MCVs should be scheduled to charge EVs before the PVs exhaust their power and can no longer function properly. Moreover, EVs consuming energy for training purposes require basic energy replenishment; at the end of the period, the remaining power of the PVs must reach the expected value set at the beginning. However, if an MCV charges a PV too early, the power consumed by subsequent training will make the final energy state lower than the

user's expectation. Therefore, the MCV must arrive at the PV after a certain time.

To address the aforementioned issues, we model a parked-vehicle-assisted edge computing network, where both PV selection and charging scheduling are considered. We aim to minimize the overall energy consumption of the modeled vehicular network while satisfying the training task requirements and customers' charging demands. In the considered model, a set of parked EVs are selected to perform the training task. Based on the task allocation strategy, the training power consumption combined with the power expectation of EV users forms the final charging demand. An MCV is dispatched to arrive at EV parking locations for energy replenishment within their required time windows. To solve this coupled optimization problem, a deep reinforcement learning (DRL)-based method is proposed. Moreover, we integrate a marginal approximation problem to solve the coupling relationship between the two optimization problems.

The main contributions of this paper can be summarized as follows:

- 1 A joint task allocation and MCV scheduling optimization problem is modeled in which the aim is to minimize the task energy consumption and MCV traveling cost. This problem is subjected to the energy limitation of MCVs and arrival time constraints.
- 2 To solve the joint optimization problem, a DRL-based algorithm combined with marginal approximation is proposed to jointly optimize the task allocation and MCV scheduling.
- 3 The effectiveness of the proposed algorithm is verified through simulation experiments and comparison with other methods. The proposed algorithm significantly reduces the network's overall energy consumption.

The remainder of this paper is structured as follows. Sect. 2 describes the system model and presents the problem description, including network, energy consumption and charging model. In Sect. 3, a joint optimization algorithm is proposed. In Sect. 4, the numerical results and analysis are presented. Finally, conclusions are drawn in Sect. 5.

2 System Model

2.1 Network Model

In this network model, We consider a parked vehicular network composed of a set of PVs $\mathcal{PV} = \{pv_1, pv_2, \dots, pv_m\}$, a cloud server that is responsible for publishing the tasks, and an MCV that is dispatched from the depot to travel along the scheduled charging tour.

In a certain area, charging customers leave their EVs at parking places. We assume that in the case of long-term EV parking, such as when the owner is working, there is no strict time limit for charging requirements. Moreover, only long-term parking makes it possible for vehicles to perform FL tasks since computation processes generally take a considerable amount of time.

Based on the task framework in [4], at the beginning of a joint optimization period, the cloud server releases one task, and each PV uses only local data to complete the task. D_i is the size of the local data, and f_i is the CPU frequency of PV i . Moreover, the earth mover distance (EMD) [5] is used to describe the effect of the PV's local data distribution on the overall task utility, which is denoted as v_i .

The cloud server must choose the optimal subset of PVs $X \subseteq PV$ to complete the task to minimize the total energy consumption while ensuring overall task utility. Once a PV is recruited, the cloud server sends it the parameters of model M_j . Client i trains the model on its local data D_i . After training, the client transmits the trained local model to the server.

The global model quality of a task can be modeled by the total data size and average EMD value of the selected PVs set:

$$U(X) = \beta(\gamma) - \kappa_1 e^{-\kappa_2(\kappa_3 D)\beta(\gamma)}, \quad (1)$$

where D is a function of the total data size of the selected PVs, γ is the average EMD value, and $\beta(\gamma) = \kappa_4 \exp(-(\frac{\gamma + \kappa_5}{\kappa_6})^2)$ is a function of the average EMD value.

2.2 Energy Consumption Model

Once the task allocation decision is completed, the cloud server sends the parameters of the model to the distributed PVs, and the transmission process consumes energy. Similarly to [6], we define the achievable data rate of PV i as:

$$r_i = B \log_2 \left(1 + \frac{Ph}{N_0 B} \right), \quad (2)$$

where B is the bandwidth available to the cloud server, P is the unified transmission power, h is the channel gain, and N_0 is background noise. The data size of the model is M , so the power consumed to transmit the model is

$$E_i^{tra} = P \frac{M}{r_i}. \quad (3)$$

Moreover, the local computation of the FL model consumes a substantial amount of energy. The CPU energy cost per second is $\rho f_i(t)$ [7], which is related to the chip architecture of the CPU frequency and $f_i(t)$ is the CPU frequency of PV i . We assume that each PV performs τ local model updates. Thus, the total energy consumed in the computation process and the consumption rate of PV i which is selected are defined as

$$\begin{aligned} E_i^{com} &= \tau D_i H_i \rho f_i^2, \\ e_i^{com} &= \rho f_i, \end{aligned} \quad (4)$$

where H is the number of CPU cycles required to compute one sample of a PV's local data and τ is the number of local training rounds.

Given the selected PV set X , the total energy consumption for the task can be defined as

$$E^{task}(X) = \sum_{i \in X} (E_i^{tra} + E_i^{com}). \quad (5)$$

2.3 Charging Model

The central controller collects a bundle of charging demands from parked EVs. For PV i , the charging demand is denoted as $pv_i = \{e_i^{ful}, e_i^{ini}, e_i^{exp}, x_i, y_i\}$, $i \in \{1, 2, \dots, m\}$, where e_i^{ful} , e_i^{ini} , e_i^{exp} and x_i, y_i denote the battery capacity, remaining energy, expectation power state and location in the 2D plane, respectively. To simplify the model, only one MCV is considered. An MCV with energy storage E^{MCV} is dispatched from the depot located at the center of the model area. As the MCV travels along the set route, once its power is exhausted, the MCV will return to the depot to replenish its power. We assume that each PV is charged only once during a single period. At the end of this stage, the power state of all PVs must reach the expected value e_i^{exp} . For PVs that are not selected to the perform training task, since their state of charge (SOC) is stable, the MCV can reach their parking location at any time. Thus, the amount of energy and charging time of these PVs can be calculated as:

$$\begin{aligned} E_i^{char} &= e_i^{exp} - e_i^{ini}, \forall i \notin X, \\ t_i^{char} &= \frac{e_i^{exp} - e_i^{ini}}{e_i^{char}}, \forall i \notin X. \end{aligned} \quad (6)$$

The SOC of PVs that are selected decreases with time, which means the task energy consumption must be considered. Since the amount of energy consumed for communication is small compared to that for computation, it can be ignored in the following equations.

$$\begin{aligned} E_i^{char} &= E_i^{com} + e_i^{exp} - e_i^{ini}, \forall i \in X, \\ t_i^{char} &= \frac{E_i^{com} + e_i^{exp} - e_i^{ini}}{e_i^{char}}, \forall i \in X. \end{aligned} \quad (7)$$

The charging sequence vector is expressed as $\alpha(PV, X) = [\alpha_0, \alpha_1, \alpha_2, \dots, \alpha_j, \alpha_T]$, where α_j represents the j_{th} visiting PV. Moreover, we assume that the MCV starts from the depot and returns to it after charging. Thus, $\alpha_0 = \alpha_T = 0$. To reduce the complexity of the model, each PV is visited only once, which means $\alpha_j \neq \alpha_{j'}$ for $j \neq j'$, $\alpha_j \neq 0, \alpha_{j'} \neq 0$. Notably, the arrival time of the MCV for each PV depends on the traveling and charging time on previous routes; the arrival time is denoted as t_i^a and calculated as:

$$t_i^a = \sum_{j=1}^{\alpha_{j+1}=i} \frac{dist(\alpha_j, \alpha_{j+1})}{v} + \sum_{j=1}^{\alpha_{j+1}=i} t_{\alpha_j}^{char}, \forall i \in \{1, 2, \dots, T\}, \quad (8)$$

where v is the moving speed of the MCV and $dist(\alpha_j, \alpha_{j+1})$ is the distance between pv_{α_j} and $pv_{\alpha_{j+1}}$. Specifically, we assume that the MCV uses battery swapping technology, so the time spent in the depot can be ignored. Therefore, $t_0^{char} = 0$.

In practice, once the PV's SOC is lower than the minimum e_i^{min} , the PV cannot remain in operation, leading to the task failure. Thus, to ensure that all selected PVs can successfully complete the training tasks, the MCV must arrive at the PV's parking place before it runs out of power. Moreover, since the amount of charging energy is fixed and the SOC of a PV depends on the arrival time, if the MCV arrives at a PV too early, the subsequent training power consumption may make the final power state lower than the expected value of the PV, even if it is fully charged. Therefore, MCV scheduling is transformed into a sequential decision problem with time window constraints. We denote the time window of each selected PV i as $tw_i = \{tw_i^1, tw_i^2\}$. Based on the initial SOC e_i^{ini} and energy consumption rate e_i^{com} , tw_i^2 can be calculated as:

$$tw_i^2 = \frac{e_i^{ini} - e_i^{min}}{e_i^{com}}. \quad (9)$$

For tw_i^1 , we need to ensure that the SOC will not exceed the battery capacity during the process of charging the specified amount of power e_i^{char} . The inequality is denoted as:

$$(e_i^{ini} - t_i^a \times e_i^{com}) + t_i^{char} \times (e_i^{char} - e_i^{com}) \leq e_i^{ful}. \quad (10)$$

Hence, tw_i^1 can be calculated as:

$$tw_i^1 = \frac{E_i^{com} + e_i^{exp} - e_i^{ful}}{e_i^{com}} - \frac{E_i^{com} + e_i^{exp} - e_i^{ini}}{e_i^{char}}. \quad (11)$$

Given the charging tour α , the energy consumed by the MCV for traveling can be formulated as:

$$E^{travel}(\alpha) = \gamma (dist(\alpha_j, \alpha_{j+1})), \quad (12)$$

where γ represents the amount of electricity consumed by the MCV per unit of driving distance.

2.4 Problem Formulation

In this part, we describe the formulation of the joint task allocation and MCV scheduling problem. We aim to minimize the total energy consumption of the network while satisfying the requirements of the task and the charging demand of PV customers. The energy consumption includes task computation energy and MCV traveling cost. Since the charging cost depends on the demand of

PVs, which is not related to the strategy, it is ignored in the target formulation. Therefore, the total energy consumption can be formulated as:

$$E^{total}(X, \alpha) = E^{task}(X) + E^{travel}(\alpha). \quad (13)$$

The strategy of this problem consists of the optimal PV selection and optimal charging sequence, which aim to minimize the total energy consumption. Moreover, this problem is subject to task and PV charging satisfaction constraints. This joint optimization problem can be expressed as follows:

$$\min_{X, \alpha} E^{total}(X, \alpha), \quad (14)$$

$$\text{s.t.}, \alpha_k \in \{0, 1, 2, \dots, m\}, \forall k \in \{1, 2, \dots, T-1\}, \quad (15)$$

$$\alpha_0 = \alpha_T = 0, \quad (16)$$

$$\alpha_k \neq \alpha_{k'}, \forall k, k' \in \{1, 2, \dots, T-1\}, k \neq k', \alpha_k \neq 0, \alpha_{k'} \neq 0, \quad (17)$$

$$tw_i^1 \leq t_i^a \leq tw_i^2, \forall i \in \{1, 2, \dots, m\} \quad (18)$$

$$X \subseteq PV, \quad (19)$$

$$U(X) \geq \Gamma, \quad (20)$$

where constraints (15), (16) and (17) state that the MCV starts from the depot, visits all PVs only once and returns to the depot at the end; constraint (18) indicates that the arrival time of the MCV is always within the corresponding time window of the PV; constraint (19) means that the PV that completes the task is selected from the set of PVs to be charged; constraint (20) ensures that the final quality of the model trained by the selected PVs meet the requirements of the task.

3 Joint Optimization Algorithm

We can divide the problem (14) into two subproblems. For the upper layer, the cloud server chooses PVs to perform the task based on the PV location and hardware device information and then transmits the parameters of the model. The selected PVs train the model in a distributed manner, and eventually the cloud server aggregates an overall model that must satisfy the defined criteria. Therefore, the upper layer can be formulated as a 0–1 programming problem, which is NP-hard. The lower layer determines the MCV charging sequence with the time window limitations. Since the charging is determined by demand and task consumption, this subproblem can be seen as a classic vehicle routing problem (VRP), which is also NP-hard. Thus, it is very difficult to solve problem (14) directly. These two subproblems are proved to be NP-hard after analysis and tightly coupled. Based on the MCV scheduling of the lower layer, PVs can only be selected arbitrarily from the set regardless of the remaining power and charging satisfaction. Meanwhile, based on the assignment of the FL task in the upper layer, the MCV can be scheduled according to the calculated time windows. In the following section, a DRL-based algorithm for the charging scheduling problem is introduced. Then, based on the solution of the lower layer, a joint optimization algorithm is designed.

3.1 DRL Algorithm for the Mobile Charging Scheduling Problem

The mobile charging scheduling problem is described as a DRL model that uses a pointer network and an actor-critic training approach. similar to [8]. We will define the Markov decision processes (MDPs) and the training procedures of the model in the following part.

The state vector of MDPs is defined as $X_t = \{x_t^i, i = 0, \dots, m, t = 0, 1, \dots, T\}$, which changes over time. Moreover, the i -th PV's state vector in time step t is formulated as $\{x_t^i = (s_t^i, d_t^i)\}$, which includes the static and dynamic state elements. The static elements s_t^i are defined as PV coordinates $\{x_i, y_i\}$ that remain constant throughout the decision-making process. The dynamic factors d_t^i include the PV charging requirement $demand_t^i$ and the MCV's power state $storage_t^i$. The next destination of the MCV in the tour affects the demand of PVs and the remaining energy of the MCV itself. Therefore, the dynamic elements of the state change between the decoding steps. When a PV is visited, its charging demand drops to zero in next time step, and the MCV subtracts the corresponding power. When the MCV's battery is depleted and it is scheduled to travel to the next PV, it will return to the depot to recharge its own battery before proceeding.

The state transfer process can be defined based on the state vector. In each time step t , with the state input X_t , the output decision vector α_{t+1} is given by the neural network after decoding which value belongs to the input index, which represents the next destination. Starting with the initial state X_0 , the state will gradually change to the termination state X_T when all PV charge requests have been met. During this process, the charging sequence α can be given. Moreover, the neural network's output is actually a set of probabilities, each of which represents the likelihood that a specified ordinal number will be selected as the next visiting point. Then, the state transition equation can be defined as $X_{t+1} = f(\alpha_{t+1}, X_t)$. The probability of the entire charging sequence is denoted as $P(\alpha|X_0) = \prod_{t=0}^T P(\alpha_{t+1}|A_t, X_t)$, where A_t represents the charging sequence up to time step t . The goal of training is to improve the probability of the route that minimizes the traveling energy consumption while satisfying the time windows constraints. For time windows, since the existing DRL research does not have a reasonable solution for hard constraints, we add a penalty to the reward function to ensure the MCV arrives at each PV within the time window, if feasible. Therefore, the reward function can defined as $R = \sum_{j=0}^m \gamma(dist(\alpha_j, \alpha_{j+1})) + \sum_{t=1}^T p_i^{exc}$, where p_i^{exc} is the seconds outside of the time windows.

Since the order of the state input is not meaningful when training the neural network, the embedded inputs replace the RNN hidden states in the encoder to reduce the computational complexity. Static and dynamic elements are mapped to a D -dimensional vector space using two one-dimensional (1D) convolution operations. Specifically, the dynamic elements must be embedded in each time step, while the static elements are processed only once. The embedded input i is denoted as $\bar{x}_t^i = (\bar{s}_t^i, \bar{d}_t^i)$. Since the position of the MCV is a time-related

sequence, it is further processed using a gated recurrent unit (GRU) to obtain the vector \bar{r}_t .

After processing all the inputs, the final output is obtained by decoding with the attention mechanism. Through the attention mechanism, the relevant information of each input data with the next decoding output can be measured and denoted as a_t . The vector a_t is computed as:

$$a_t = \text{softmax}(v_a^T \tanh(W_a[\bar{x}_t^i; h_t])), \quad (21)$$

where h_t is the memory state of the GRU. Then, by combining the attention vector and state input, the context vector can be calculated as $c_t = \sum_{i=0}^m a_t^i \bar{x}_t^i$. Finally, we denote $P(\alpha_{t+1}|A_t, X_t) = \text{softmax}(v_c^T \tanh(W_c[\bar{x}_t^i; c_t]))$ as the conditional probability of the next-visited PV. In the training stage, we use a random strategy to sample PVs based on probabilities, while in the validation and inference stages, a greedy strategy is adopted.

To train this network, the policy gradient method REINFORCE [9] is used. In addition to the actor network, a critic network used to estimate the reward function based on the initial state is included. We denote the training dataset as θ , which includes the VRP instances with time windows following a probability distribution Φ_θ . We sample k problems from the dataset in each training epoch and use the actor network to generate a feasible charging sequence. Based on the reward function and charging tour, the corresponding reward can be computed. In epoch i , the reward approximation $V(X_0^i)$ can be calculated by the critic network and then used in the policy gradient to accelerate the training of the actor network. Moreover, the observed rewards can be used to update the parameters of the critic network and shift its estimated reward value based on the state closer to the true value.

3.2 A Joint Optimization Algorithm Based on the Marginal Product Formula

Based on the solution of the lower layer, a joint optimization algorithm based on the marginal product formula is proposed. The core concept of this algorithm is that for the set of PVs selected, we choose the PV with the greatest gain per unit cost until the requirements of the training task are satisfied. The cost here includes the computational energy consumption and the MCV's travel energy consumption; the previous part can be computed directly by Eq. (5) and the latter is inferred from the DRL model. Therefore, the marginal product formula based on the selection result sets X in each round for each PV i can be defined as

$$M(X, i) = \frac{U(X \cup i) - U(X)}{E(X \cup i, \alpha) - E(X, \alpha)}. \quad (22)$$

Based on the marginal product formula, we continuously add the PV with the maximum $M(X, i)$ to the set X until the utility requirements of the task are met. The details of the algorithm are shown in Algorithm 1.

Algorithm 1. Joint Task Allocation and Charging Scheduling

Input: PV, τ .

Output: X, α . **Phase I - Initialization:** Initialize data: $X = \emptyset, \alpha = \emptyset, E^{task}(X) = 0, E^{travel}(X) = 0, U(X) = 0$.

- 1: **while** $U(X) \leq \tau$ **do**
 - 2: **for** $i \in PV$ **do**
 - 3: $e_i^{char} = E_i^{com} + e_i^{exp} - e_i^{ini}$
 - 4: $tw_i^1 = \frac{E_i^{com} + e_i^{exp} - e_i^{ful}}{e_i^{com}} - \frac{E_i^{com} + e_i^{exp} - e_i^{ini}}{e_i^{char}}$
 - 5: $tw_i^2 = \frac{e_i^{ini} - e_i^{min}}{e_i^{com}}$
 - 6: Use trained DRL model to obtain the charging tour α ; the arrival time of the MCV must satisfy the time window. When the remaining power of the MCV is not sufficient to reach the next destination and then return to the depot, the MCV will return to the warehouse first to replenish its energy.
 - 7: **end for**
 - 8: $best = \arg \max_{i \in PV} \frac{U(X \cup i) - U(X)}{E(X \cup i, \alpha) - E(X, \alpha)}$
 - 9: $X = X \cup best$
 - 10: $PV = PV \setminus best$
 - 11: **end while**
 - 12: **return** $X, \alpha(PV, X)$
-

4 Simulation Results and Analysis

In this section, we measure the effectiveness of the proposed algorithm by means of numerical experiments and comparison with other methods. Moreover, the effect of MCV capacity is evaluated. The detailed experimental parameters are presented in Table 1. The parameters in Eq. (1) is set based on [5]. We train the DRL model for 20 epochs on a dataset with 120,000 instances. The batch and validation sizes are set to 200 and 1,000, respectively. The compared methods include the greedy algorithm and reward-cost ratio (RC-Ratio) method [10]. In the greedy algorithm, the PV that has the maximum utility is selected in each

Table 1. Main Experiment Parameters

Parameter	Value	Parameter	Value
m	20	e_i^{char}	60 kw
E_i^{ful}	50 kwh	MCV Capacity	200 kwh
E_i^{exp}	20 kwh	f_i	[1.6, 2] GHZ
E_i^{ini}	[3, 9] kwh	τ	15
E_i^{min}	3 kwh	D_i	(5, 10]
Area size	36 km × km	H	[1, 3] cycles/sample
v	10 m/s	ρ	0.5
γ	500 j/m	EMD_i	[0, 1.2]

round until the task target utility is satisfied. For the charging scheduling tour, the MCV selects the PV with the earliest deadline as the next destination. Once all PVs with time windows are visited, the MCV drives to the nearest PV. For the RC-Ratio method, the PV with the maximum marginal value is added to the PV set to perform the training task, and the corresponding charging tour is calculated in the same way as it is in the greedy algorithm.

Figure 1 compares the energy consumption of the entire network, which is the objective function in the modeled joint optimization problem. There is a positive correlation between energy usage and target utility because as the target utility increases, more PVs are selected to perform the training task, thereby consuming a large amount of energy. Moreover, the selection of the upper PVs imposes more time window constraints on the lower MCV scheduling, resulting in an increase in the traveling cost. The proposed algorithm outperforms the two other methods because the greedy algorithm focuses on maximizing the benefits and ignores the computational energy consumption of different PVs and the impact on the lower-level MCV scheduling. Although the RC-Ratio method chooses the PV with the maximum marginal product, the MCV scheduling algorithm called at the lower layer is less effective than our trained DRL model. Therefore, our proposed algorithm, which combines the marginal product with DRL, achieves the best performance. To illustrate the performance of the DRL method, Fig. 2 compares the MCV efficiency of the three algorithms with different target utility. The MCV efficiency is defined as the percentage of charging energy for PVs to the total energy consumption of the MCV. The simulation results show that the proposed algorithm outperforms the greedy and RC-Ratio methods. On the one hand, the trained DRL minimizes the traveling cost with the time window constraints, which outperforms the other two methods. On the other hand, the proposed algorithm jointly selects PVs and determines the charging sequence, which reduces the time constraints from a global perspective. However, the greedy and RC-Ratio methods ignore the effect of selection on the charging scheduling. Meanwhile, these two algorithms consider only the time window requirements and do not attempt to minimize the overall travel consumption. Therefore, the proposed algorithm can improve the energy utilization and reduce energy waste.

In Fig. 3, we study the effect of MCV capacity on network energy consumption. As the capacity of the MCV increases from 100 kWh to 200 kWh, the overall energy consumption of the network decreases because the increased MCV capacity reduces the number of journeys back to the depot to refuel, resulting in lower travel energy usage. Moreover, the DRL model can be trained to make better decisions in each stage of selecting a destination to visit, thereby reducing the energy consumed for traveling. Therefore, the proposed algorithm outperforms the greedy and RC-Ratio methods.

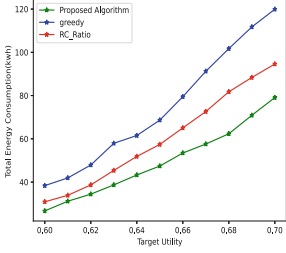


Fig. 1. Overall network energy consumption with different target utility

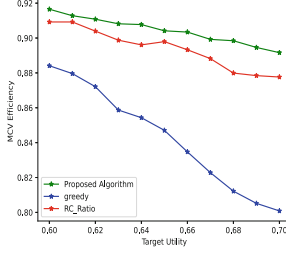


Fig. 2. MCV efficiency with different target utility

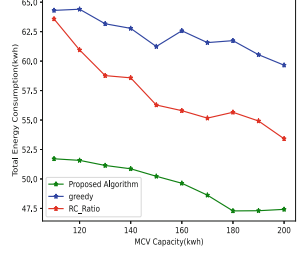


Fig. 3. Overall network energy consumption with different MCV capacity

5 Conclusions

In this paper, we consider a joint task allocation and charging scheduling problem in a parked-vehicle-assisted edge computing network. To minimize the overall energy consumption of this modeled network with task training target limitations and PV energy level expectations, we design a joint optimization algorithm that combines the DRL method with the marginal product concept. The experimental results verify that the proposed algorithm can effectively reduce the overall energy consumption of the network.

References

1. Chen, H., Su, Z., Hui, Y., Hui, H.: Dynamic charging optimization for mobile charging stations in internet of things. *IEEE Access* **6**, 53509–53520 (2018)
2. Wang, H., Wang, R., Xu, H., Kun, Z., Yi, C., Niyato, D.: Multi-objective mobile charging scheduling on the internet of electric vehicles: a DRL approach, pp. 01–06 (2021)
3. Posner, J., Tseng, L., Aloqaily, M., Jararweh, Y.: Federated learning in vehicular networks: opportunities and solutions. *IEEE Network* **35**(2), 152–159 (2021)
4. Nour, B., Cherkaoui, S., Mlika, Z.: Federated learning and proactive computation reuse at the edge of smart homes. *IEEE Transactions on Network Science and Engineering* (2021)
5. Cheng, Z., Min, M., Liwang, M., Gao, Z., Huang, L.: Joint client selection and task assignment for multi-task federated learning in MEC networks, pp. 1–6 (2021)
6. Xu, B., Xia, W., Zhang, J., Sun, X., Zhu, H.: Dynamic client association for energy-aware hierarchical federated learning, pp. 1–6 (2021)
7. Luo, S., Chen, X., Wu, Q., Zhou, Z., Yu, S.: HFEL: joint edge association and resource allocation for cost-efficient hierarchical federated edge learning. *IEEE Trans. Wireless Commun.* **19**(10), 6535–6548 (2020)
8. Nazari, M., Oroojlooy, A., Snyder, L., Takác, M.: Reinforcement learning for solving the vehicle routing problem. In: *Advances in neural information processing systems* 31 (2018)

9. Bello, I., Pham, H., Le, Q.V., Norouzi, M., Bengio, S.: Neural combinatorial optimization with reinforcement learning. arXiv preprint [arXiv:1611.09940](https://arxiv.org/abs/1611.09940) (2016)
10. Wu, T., et al.: Joint sensor selection and energy allocation for tasks-driven mobile charging in wireless rechargeable sensor networks. *IEEE Internet Things J.* **7**(12), 11505–11523 (2020). <https://doi.org/10.1109/JIOT.2020.3019451>



A Secure Authentication Approach for the Smart Terminal and Edge Service

Qian He^{1(✉)}, Jing Song^{1,2}, Shicheng Wang², Peng Liu¹, and Bingcheng Jiang¹

¹ State and Local Joint Engineering Research Center for Satellite Navigation and Location Service, Guangxi Key Laboratory of Cryptograph and Information Security, Guilin University of Electronic Technology, Guilin 541004, China
heqian@guet.edu.cn

² CETC Key Laboratory of Aerospace Information Application, Shijiazhuang 050081, China

Abstract. Smart home applications make our lives more comfortable, more convenient than ever before. However, deploying smart home applications and smart terminals could pose a potential security threat to personal information and home privacy. In order to prevent illegal use of terminals and applications, it is very necessary to establish secure and reliable communication between terminal and edge server. In this paper, we design a two-party authentication and key negotiation protocol for the smart terminal and edge service. The edge-based authentication and key negotiation scheme offloads the terminal's main computational overhead to the edge side, and exploits cryptographic algorithms to achieve user anonymity and untraceability. Security is verified by the BAN logic and AVISPA. We also evaluate the performance by comparing our scheme with other related schemes in terms of computational overhead. The security and performance results show that our proposed scheme is suitable for edge-assisted smart home applications.

Keywords: Smart home · Authentication session key negotiation · Elliptic curve code · Edge service

1 Introduction

Recently, such smart home applications as anti-theft, remote monitor and help services become more and more popular [1]. These smart home applications need smart end devices to collect and upload information about the home and provide remote services, thus enhancing the convenience and intelligence of living. According to a report by the Internet Data Center, the smart home market in China has exceeded 208 million units. The huge market size of smart homes is attracting manufacturers such as Haier, Huawei, Xiaomi and Baidu to continuously develop new smart home products and keep pushing the development of smart homes and the smart home cloud also been introduced.

However, the centralized data processing method represented by cloud computing is no longer able to meet the demand for real-time and efficient data

processing at the exploding terminals due to its storage characteristics and transmission bandwidth limitations [2]. Edge computing is located close to endpoint data sources and supports low-latency services to reduce the pressure on cloud-centric processing and storage and improve real-time service responsiveness [3]. The smart home becomes an important area for edge computing applications. Judgments and action decisions are made by the edge server to the smart home, following the principle of processing data at the source, which can ensure the security of terminal data to a certain extent.

While enjoying the convenience brought by the smart home, there are also many security issues. Once a smart terminal connects to the network, it can be subject to various types of attacks, which can not only cause leakage of important information, but also threaten personal safety and property security. In January 2020, a major security vulnerability was discovered in the Xiaomi Mi Home security camera, which allowed users to view other users' home screens when connecting the camera to Google Nest Hub, which seriously breaks users' privacy.

Therefore, ensuring secure communication in the smart home is an important consideration [4]. The application security challenges faced by smart homes are divided into authentication and communication confidentiality. Authentication and key negotiation technology, as the first line of defence for smart home security communication, can achieve the integrity, confidentiality and anonymity of communication data and solve the problem of privacy leakage between smart home users and edge nodes, and the session key negotiated through authentication and negotiation is used to encrypt the data in the upload and despatch process. The focus should be on making optimisations in terms of flexibility and efficiency, designing a secure authentication key system and achieving goals such as anonymity and security for smart home users.

1.1 Related Work

An authentication and key negotiation protocol is the first protection mechanism to ensure the secure transmission of smart terminal data in the edge computing environment, which has been partially investigated by researchers. An analysis of a class of lightweight authentication and key negotiation protocols reveals that many researchers have ignored the most fundamental security in the pursuit of lower computational cost [5]. Geetha et al. [6] implement session keys using low-cost computational tools and message authentication codes, but each communication requires the help of a gateway and the communication overhead is high. Wazid et al. [7] propose a three-factor anonymous authentication protocol for smart homes based on symmetric and hash functions. It is shown that protocol security is generally proportional to the complexity of the computational tools used.

Mishrad et al. [8] scheme requires the participation of a third party authentication center, based on which the authentication center issues signature credentials as authentication credentials for all parties, but there is no guarantee that the third party is honest and trustworthy. Lyu et al. [9] proposes an elliptic

curve-based key negotiation protocol that implements authentication and key management, but the drawback is that it requires a large cost to establish a trusted third-party key distribution center to manage public and private keys. The protocol of Alshahrani et al. [10] in 2019 does not provide strong confidentiality for smart homes and is not resistant to known key attacks and denial of service attacks.

Shuai et al. [11] propose an authentication scheme using elliptic curve cryptography for effective anonymity, and in 2020 Soumya et al. [12] point out the existence of impersonation attacks and insider attacks in this scheme, and proposed the use of biometric and hash functions for lightweight authentication. However, the Soumya solution using only hash functions is subject to internal privilege attacks and forgery attacks, and anonymity is not guaranteed. Jia et al. [13] use the identity signature issued by the certification center as a temporary public key in the edge-end authentication process, combined with the Diffie-Hellman idea to guarantee the anonymity of the endpoint, but the bilinear pair operation of the scheme is not applicable to resource-constrained home endpoint devices. Li et al. [14] propose a new anonymous identity-based security scheme based on identity cryptography under the edge-end architecture, but the scheme suffers from offline dictionary attacks and lack of perfect forward security.

In summary, the current authentication and key negotiation protocols for smart homes have the following problems: First, the current mutual authentication and key negotiation protocols for smart home terminals and edge servers still require the participation of the cloud center and do not really implement edge-side authentication. Second, smart home terminals are resource-constrained devices, and some existing schemes use algorithms such as bilinear pairing, which are not applicable to resource-constrained terminals. Third, some only use hash functions to complete authentication and key negotiation, which is low in computation but lacks security. Fourth, some smart home authentication and key negotiation protocols require a trusted third party to act as an authentication center to issue signature credentials, but a true trusted third party is difficult to implement in real scenarios.

1.2 Contributions

To address the above issues, the main contributions of this paper are as follows.

- (1) To address the problem that traditional authentication and key negotiation protocols for smart homes in edge environments still require third-party assistance in authentication from the cloud center, a new authentication architecture for smart homes in edge environments is considered to provide secure and efficient communication between resource-constrained home terminals and edge servers in the event of cloud link disconnection.
- (2) We design a new edge-based authentication and key negotiation scheme AKES-TE in smart home environments, which achieves efficient authentication of home terminals and edge servers in a single round of message exchange, shifts the main computational overhead of authentication and

key negotiation to the edge side for completion, eliminates the need for the cloud center to assist in authentication in real-time, and utilizes cryptographic algorithms to process endpoint identities to achieve user anonymity and untraceability.

- (3) A proof of the security of the scheme is given using the BAN method and is also verified by simulation using the widely used formal analysis tool AVISPA, which shows that the scheme is secure. The scheme was compared with the previous method and the results showed that the new scheme guarantees a lower computational complexity without sacrificing the security objective.

The rest of the paper is structured as follows, the authentication architecture for the smart home in Sect. 2, the details of the scheme in Sect. 3, the proof of safety in Sect. 4, and the experiments comparing the security and computation performance of the scheme in Sect. 5, and the conclusion in Sect. 6.

2 Authentication Architecture

The authentication architecture of the smart home in the edge computing environment is shown in the Fig. 1. The smart terminal is the authentication user, and the edge server is the local server closest to the smart home. In the whole system, the smart home terminal can transmit data to the edge server for processing and short-term storage, the edge server can offload the computing tasks of the remote cloud server, and the edge server can make intelligent judgment and action decisions according to the data processing near the data source.

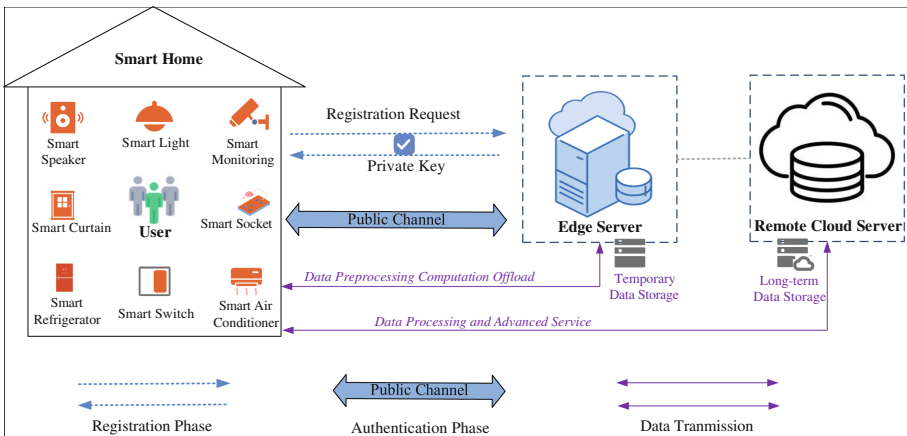


Fig. 1. Authentication architecture for the smart terminal and edge service in the smart home.

3 Proposed Scheme

Our authentication scheme for the smart home edge service consists of three phases: the registration phase, the login authentication phase, the password change phase. In the scheme, the counter CTR_S for the edge server ES_j and the counter CTR_SC for the terminal users are set to track the number of consecutive failed attempts for login and authentication, with the initial value set to 0 and the maximum failure threshold set to n . The communication process is terminated immediately when $CTR_S > n$ or $CTR_SC > n$. The symbols and descriptions used in this paper are shown in Table 1.

The authentication process works between the smart terminal (user) and the edge server. First, the user sends a registration request to the edge server and then obtains the private key. Then in the public channel, through a single round of message exchange, both parties complete the authentication and key negotiation process, and the negotiated session key is used for subsequent data communication.

Table 1. Description of symbols.

Notations	Descriptions	Notations	Descriptions
TM_i	i -th smart home user TM_i	$SCID$	Smart card identifier
ES_j	j -th edge server ES_j	p	A very large prime number
UID_i	Identification of smart home users TM_i	k	Secret keys for edge servers
G	A generating element of the group E_p	T_n	Current Timestamp
SID_j	Identity for edge servers ES_j	SC	Smart card
E_k	Symmetric encryption of key k	\parallel	Concatenation operations
D_k	Symmetric decryption of key k	\oplus	Bitwise xor operations

3.1 Registration Phase

Step 1. The registration phase is mainly for the smart home user TM_i to register with the edge server ES_j , where TM_i obtains a SC issued by ES_j and thus obtains a proof of legal identity. TM_i selects the identity UID_i , the password PW_i and the random number o_i , calculates RID_i and sends $\{RID_i, UID_i\}$ to ES_j , where RID_i is a very large number belonging to Z_p output by $H()$, which can be regarded as a string of length $|Z_p|$.

$$RID_i = H(UID_i \parallel PW_i \parallel o_i) \tag{1}$$

Step 2. The ES_j receives $\{RID_i, UID_i\}$ and checks the validity of the UID_i to determine if the TM_i has already been registered. If the TM_i has already been registered, the ES_j returns a message for the TM_i to reselect a new identity for registration. If the TM_i 's identity has not been registered before, the smart card identifier $SCID$, identity SID_j and o_j are selected and then $\{ID, EID, V_0^j\}$ is

calculated. Finally, $\{ID, SCID, CTR_S\}$ is stored in the database, $\{EID, V_0^j\}$ is written to the SC and the SC is sent to the TM_i via a secure channel.

$$ID = H(UID_i \parallel SID_j \parallel SCID) \quad (2)$$

$$EID = E_k(ID \parallel o_j), V_0^j = H(ID \parallel k) \oplus RID_i \quad (3)$$

Step 3. When the TM_i receives the SC , it initializes the CTR_SC to 0 and updates the $SC \rightarrow \{EID, V_0^j, CTR_SC\}$.

3.2 Login and Authentication Phase

The smart home user and the edge server should authenticate with each other and negotiate a session key for subsequent secure communication after the authentication is completed. The specific steps for the negotiation of the certificate and key are shown below.

Step 1. When the TM_i wants to establish a connection with the ES_j for data communication, the TM_i needs to verify the validity of the identity and password. the TM_i enters the UID_i and PW_i and if it is wrong then $CTR_SC = CTR_SC + 1$. if it is correct and $CTR_S < n$, the SC selects the random number $r_i \in Z_p$, calculates $\{X_u, X_u^1, V_1^i\}$ and sends $\{V_1^i, EID, X_u^1, T_1\}$ to the ES_j .

$$X_u = r_i \cdot G, X_u^1 = X_u \oplus H(V_0^j \parallel RID_i) \quad (4)$$

$$V_1^i = H(V_0^j \parallel RID_i \parallel T_1) \quad (5)$$

Step 2. After the ES_j receives $\{V_1^i, EID, X_u^1, T_1\}$, it first checks the time validity, determines whether $|T_1^s - T_1| < \Delta T$ holds, ΔT being the time delay threshold of the ES_j . Then decrypts the $(ID \parallel o_j) = D_k(EID)$ to check the legitimacy of the TM_i . If the ID is incorrect, it calculates $CTR_S = CTR_S + 1$ and terminates the communication session. If the TM_i is legitimate and $CTR_s < n$, the calculation determines whether V_1^j is equal to V_1^i . If equal then a random number r_j and b_s are chosen to calculate $\{X_u, X_s, X_s^1, EID_{new}, V_2^j, V_3^j\}$ and finally $\{V_2^j, V_3^j, X_s^1, T_2\}$ is sent to the TM_i , where T_2 is the current timestamp.

$$V_1^j = H(V_0^j \parallel RID_i \parallel T_1), X_u = X_u^1 \oplus H(V_0^j \parallel RID_i) \quad (6)$$

$$X_s = r_j \cdot G, X_s^1 = X_u \oplus X_s \quad (7)$$

$$EID_{new} = E_k(ID \parallel b_s), V_2^j = H(X_s) \oplus EID_{new} \quad (8)$$

$$V_3^j = H(X_s \parallel EID_{new} \parallel T_2) \quad (9)$$

Step 3. When TM_i receives $\{V_2^j \parallel V_3^j \parallel X_s^1 \parallel T_2\}$, it first verifies the time validity, calculates $\{X_s^u, EID_{new}^u, V_3^i\}$ if $|T_2^u - T_2| < \Delta T$ holds, and determines if $V_3^i = V_3^j$ holds. If not, $CTR_SC = CTR_SC + 1$, the TM_i denies the ES_j access and terminates the session. If equal then continue to compute $\{V_4^i, SK\}$ and finally return $\{V_4^i, T_3\}$ to the ES_j , where T_3 is the current timestamp of the TM_i , while initialising CTR_SC to 0.

$$X_s^u = X_s^1 \oplus X_u, EID_{new}^u = V_2^j \oplus H(X_s^u) \quad (10)$$

$$V_3^i = H(X_s^u \parallel EID_{new}^u \parallel T_2), V_4^i = H(X_u \parallel V_1^i \parallel X_s^u \parallel T_3) \quad (11)$$

$$SK = H(X_s^u \parallel X_u \parallel UID_i \parallel SID_j) \quad (12)$$

Step 4. Upon receipt of $\{V_4^i, T_3\}$, the ES_j similarly first verifies that T is satisfied. If not, the ES_j denies service and terminates the session, setting $CTR_S = CTR_S + 1$. If satisfied, it computes V_4^j and determines if it is equal to V_4^i , which is used to determine the user's legitimacy. When they are equal, the session key SK is calculated and CTR_S is set to 0. This determines that the ES_j and TM_i authentication is successful and that the session key between the ES_j and TM_i is SK .

$$V_4^j = H(X_u \parallel V_1^j \parallel X_s \parallel T_3) \quad (13)$$

$$SK = H(X_s \parallel X_u \parallel UID_i \parallel SID_j) \quad (14)$$

3.3 Password Change Phase

When the TM_i wants to change the password, it first enters the old PW_i to calculate the $RID_i = H(UID_i \parallel PW_i \parallel o_i)$ and then sends it to the ES_j . The ES_j simply calculates the $V_0^j_new = H(ID \parallel k) \oplus RID_i$ and if $V_0^j_new = V_0^j$, it returns that the user is allowed to change the password and calculates the $RID_i^{new} = H(UID_i \parallel PW_i^{new} \parallel o_i^{new})$ with the new password, the rest of the steps are the same as in the registration phase. Otherwise, the password change is rejected and the session is terminated.

4 Security Analysis

4.1 Proof of BAN Logic

BAN(Burrows-Abadi-Needham) logic [15] is a formal method of belief-based modal logic for analysing and verifying authentication protocols and session keys, consisting of propositional knowledge or inference rules, where the proposition represents the subject's knowledge or belief about the message.

- (1) Goals: Goal G1: $TM_i \models TM_i \xleftrightarrow{SK} ES_j$. Goal G2: $ES_j \models TM_i \xleftrightarrow{SK} ES_j$.
 Goal G3: $TM_i \models ES_j \models TM_i \xleftrightarrow{SK} ES_j$. Goal G4: $ES_j \models TM_i \models ES_j \xleftrightarrow{SK} TM_i$.
- (2) Idealized Communication Information $Mesg1 : TM_i \rightarrow ES_j : \{UID_i, RID_i\}$. $Mesg2 : ES_j \rightarrow TM_i : \{EID, V_0^j\}$. $Mesg3 : TM_i \rightarrow ES_j : \{V_1^i, EID, X_u^1, T_1\}$. $Mesg4 : ES_j \rightarrow TM_i : \{V_2^j, V_3^j, X_s^1, T_2\}$. $Mesg5 : TM_i \rightarrow ES_j : \{V_4^i, T_3\}$.
- (3) Initialisation Phase Assumptions:

$$\begin{aligned} \delta 1 : TM_i \models \#(X_s) & \quad \delta 5 : TM_i \models ES_j \mid \Rightarrow X_s \\ \delta 2 : ES_j \models \#(X_u) & \quad \delta 6 : ES_j \models TM_i \mid \Rightarrow X_u \\ \delta 3 : TM_i \models \#(r_i) & \quad \delta 7 : TM_i \models TM_i \xleftrightarrow{EID} ES_j \\ \delta 4 : ES_j \models \#(r_j) & \quad \delta 8 : ES_j \models TM_i \xleftrightarrow{H(V_0^j \parallel RID_i)} ES_j \end{aligned}$$

- (4) Major Proofs using BAN Logic Rules:

For G1, we get $\Upsilon 1 : TM_i \triangleleft \{V_2^j, V_3^j : (X_s, EID, T_2), X_s^1, T_2\}$ from Mesg4 and the reception rule. By $\delta 7$, $\Upsilon 1$ and the message meaning rule we can get $\Upsilon 2 : TM_i \models ES_j \mid \sim (X_s, T_2)$. We get $\Upsilon 3 : TM_i \models ES_j \models (X_s, T_2)$ from $\delta 1$, $\Upsilon 2$ and the provisional validation rule. We get $\Upsilon 4 : TM_i \models ES_j \models X_s$ from $\Upsilon 3$ and the belief rule. we get $\Upsilon 5 : TM_i \models X_s$ from $\delta 5$, $\Upsilon 4$ and the arbitration rule, and we get $\Upsilon 6 : TM_i \mid \equiv TM_i \xleftrightarrow{SK} ES_j$ from $SK = H(X_s \parallel X_u \parallel UID_i \parallel SID_j)$ (G1 is certified).

For G2, we can get $\Upsilon 7 : ES_j \triangleleft \{V_1^i, EID, X_u^1, T_1\}$ according to Mesg3 and the receiving rule. We can get $\Upsilon 8 : ES_j \triangleleft \{X_u\}_{H(V_0^j \parallel RID_i)}$ by $\Upsilon 7$ and receiving rules. By $\delta 8$, $\Upsilon 8$ and the message meaning rule we can get $\Upsilon 9 : ES_j \models TM_i \mid \sim X_u$. We can get $\Upsilon 10 : ES_j \models TM_i \models X_u$ by $\delta 2$, $\Upsilon 9$ and temporary value validation rules. We can get $\Upsilon 11 : ES_j \models X_u$ through $\delta 6$, $\Upsilon 10$ and Jurisdiction rules. Combined with $SK = H(X_s \parallel X_u \parallel UID_i \parallel SID_j)$ we can get $\Upsilon 12 : ES_j \models TM_i \xleftrightarrow{SK} ES_j$ (G2 certified).

For G3, according to $\Upsilon 6 : TM_i \mid \equiv TM_i \xleftrightarrow{SK} ES_j$, $\delta 3$ and session key rules, we can get $\Upsilon 13 : TM_i \models ES_j \models TM_i \xleftrightarrow{SK} ES_j$ (G3 certified).

For G4, according to $\Upsilon 13$, $\delta 4$ and provisional verification rules, we can get $\Upsilon 14 : ES_j \models TM_i \models ES_j \xleftrightarrow{SK} TM_i$ (G4 certified).

4.2 AVISPA Verifying

We simulate our protocol using the AVISPA [16] security protocol analysis tool, which allows us to check and evaluate whether the protocol AKES-TE is resistant

to various security attacks. Because of the XOR operation in AKES-TE, we have used both OFMC and CL-AtSe backends in our simulation experiments.

The simulation results are shown in Fig. 2. In the OFMC model, the total number of nodes visited is 4 and the depth is 2. In the CL-AtSe model, the number of analyzed states is 1686, of which 724 can be achieved, the transition time is 0.02s, and the computation time is 0.84s. The results show that the protocol AKES-TE is safe under the OFMC and CL-AtSe models.

<pre> % OFMC % Version of 2006/02/13 SUMMARY SAFE DETAILS BOUNDED_NUMBER_OF_SESSIONS PROTOCOL /home/span/span/testsuite/results/ Terminal_EdgeServer.if GOAL as_specified BACKEND OFMC COMMENTS STATISTICS parseTime: 0.00s searchTime: 0.01s visitedNodes: 4 nodes depth: 2 plies </pre>	<pre> SUMMARY SAFE DETAILS BOUNDED_NUMBER_OF_SESSIONS UNTYPED_MODEL PROTOCOL /home/span/span/testsuite/results/ Terminal_EdgeServer.if GOAL As Specified BACKEND CL-AtSe STATISTICS Analysed : 1686 states Reachable : 724 states Translation: 0.02 seconds Computation: 0.84 seconds </pre>
---	--

Fig. 2. AVISPA experimental simulation results.

5 Performance Analysis

5.1 Functional Comparison

The secure functions of AKES-TE are compared with these authentication protocols [5, 8, 13, 14] shown in Table 2. In Table 2, "✓" indicates that the attack can be resisted, and "×" indicates that the attack cannot be resisted. As can be seen, the new scheme addresses some serious security threats of the authentication key exchange protocol and makes countermeasures to make it more secure compared to other schemes, while using low-energy encryption to make it suitable for resource-constrained terminals. We use random number (r_i, r_j, o_i, o_j), timestamp(T_n), a reasonable one-way hash cipher algorithm and elliptic curve cipher algorithm to ensure the integrity of the session information and achieve resistance to other attacks such as replay attacks and man-in-the-middle attacks.

Table 2. Function Comparison.

Function	Chen [5]	Mishra [8]	Jia [13]	Li [14]	AKES-TE
Anonymity	✓	✓	✓	✓	✓
Resist forged counterfeit attacks	✓	✓	✓	✓	✓
Resist offline dictionary attacks	×	✓	×	✓	✓
Resist to replay attacks	✓	✓	✓	×	✓
Resist to known key attacks	✓	✓	×	✓	✓
Resist SC loss of attack	✓	✓	✓	×	✓
Resist man-in-the-middle attacks	✓	×	✓	✓	✓
Resist Dos attacks	×	✓	✓	×	✓
Enables forward security	×	×	×	✓	✓

5.2 Computing Performance Comparison

A comparison of the computational performance of this protocol with other protocols is shown in Table 3. Because the edge server and smart terminal have different computational capabilities, the operations were simulated on two different platforms (see Jia [13]). The edge server was simulated by Alibaba’s cloud platform with Intel(R) Xeon(R) CPU E5-2630 0 @ 2.30 GHz, 1 GB RAM and Ubuntu 14.04 with resource-constrained Terminal emulated by Google Nexus 4.4.2 OS with 2 GHz ARM CPU armeabi-v7a, 300 MiB RAM.

Table 3. Running time of basic operations.

Operations	User	EdgeServer
Bilinear pairing (T_B)	48.660 ms	5.275 ms
Scalar multiplication (T_S)	19.919 ms	1.97 ms
Point addition (T_P)	0.118 ms	0.012 ms
Hash function (T_H)	0.089 ms	0.009 ms
Modular exponentiation (T_M)	3.328 ms	0.339 ms
Symmetric encryption/decryption (T_{Sy})	6.6967 ms	0.65 ms

Table 4. Computing performance comparison.

Scheme	User (ms)	EdgeServer (ms)	Total (ms)
[5]	$2T_S + 12T_H \approx 40.906$	$2T_S + 8T_H \approx 4.012$	44.918
[8]	$T_M + 2T_S + 5T_H + T_{Sy} \approx 50.3077$	$T_B + 3T_S + 4T_H + T_{Sy} \approx 11.880$	62.1877
[13]	$T_M + 4T_S + 5T_H \approx 83.449$	$T_B + 6T_S + 5T_H + 3T_P \approx 17.176$	100.625
[14]	$4T_P + 6T_S + 5T_H \approx 120.431$	$T_B + 4T_S + 2T_H + 3T_P \approx 13.209$	133.64
Our	$T_S + 6T_H \approx 20.453$	$T_S + 6T_H + 2T_{Sy} \approx 3.324$	23.777

As shown in Table 4, we can see that the Chen scheme [5] has a computational cost of 44.918ms in the authentication phase, while it takes 40.906 ms on the smart terminal side. Obviously, the authentication cost of the scheme in this paper is only 23.777ms, the computing cost of the smart terminal is 20.453 ms, and the computing cost of the edge server is 3.324ms. From the table comparison, it can be seen that the calculation cost of this scheme is lower than the other four schemes, and it is more suitable for the smart home. As shown in Figs. 3, 4, and 5, the different solutions are compared in terms of computing performance at the user side, computing performance at the edge server and total computing time.

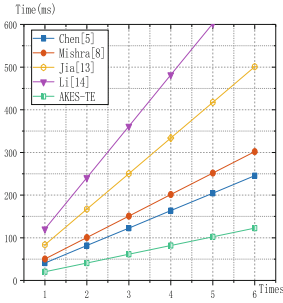


Fig. 3. Computing performance for users

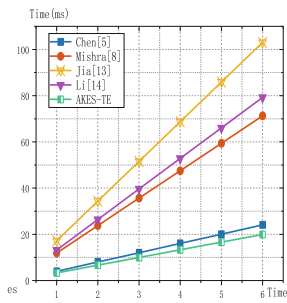


Fig. 4. Computing performance for edgeservers

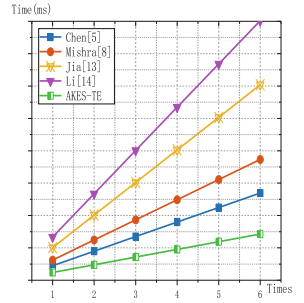


Fig. 5. Total computing performance

6 Conclusion

With the widespread use of edge computing in smart homes, this paper proposes an authentication and key negotiation protocol for smart homes in an edge environment, AKES-TE. It achieves efficient authentication between smart homes and edge servers in a single round of message exchange, and realizes edge-side authentication without the involvement of cloud servers. The main computational overhead of authentication and key negotiation is shifted to the edge side, and the end-user identity is cryptographically processed to achieve user anonymity and untraceability. Reduces endpoint energy consumption and enables lightweight authentication using symmetric encryption. Fast and efficient authentication and key negotiation are achieved using passphrases and tamper-proof smart cards, avoiding the use of public key infrastructure. Rigorous security verification and attestation are then performed using BAN logic methods and AVISPA tools. Finally, an experimental comparison shows that this scheme offers better performance in smart home security authentication.

Acknowledgements. This work was partially supported by the National Natural Science Foundation of China (62162018,61967005), Guangxi Innovation-driven Development Project (AA172020-24), and the CETC Key Laboratory of Aerospace Infor-

mation Application Foundation, and Guangxi Key Laboratory of Cryptography and Information Security Discovery (GCIS201701).

References

1. Misra, S., et al.: Blockchain at the edge: performance of resource-constrained IoT networks. *IEEE Trans. Parallel Distrib. Syst.* **32**(1), 174–183 (2020)
2. Shi, W., Dustdar, S.F.: The promise of edge computing. *Computer* **49**(5), 78–81 (2016)
3. Chen, Z., He, Q., Liu, L., Lan, D., Chung, H.-M., Mao, Z.: An artificial intelligence perspective on mobile edge computing. In: 2019 IEEE International Conference on Smart Internet of Things (SmartIoT), pp. 100–106. Tianjin (2019)
4. Hermanu, C., et al.: Dual mode system of smart home based on internet of things. *J. Robot. Control* **3**(1), 26–31 (2022)
5. Chen, C.M., et al. F.: Lightweight authentication protocol in edge-based smart grid environment. *EURASIP J. Wirel. Commun. Netw.* **21**(1), 1–18 (2021)
6. Geetha, A.V.: Threshold cryptography-based group authentication scheme for the smart home environments. *LBS Coll. Eng. Kasaragod* **5**(1) (2016)
7. Wazid, M., et al. F.: Secure remote user authenticated key establishment protocol for smart home environment. *IEEE Trans. Dependable Secure Comput.* **17**(2), 391–406 (2017)
8. Mishra, D., et al.: A provably secure dynamic ID-based authenticated key agreement framework for mobile edge computing without a trusted party. *J. Inf. Secur. Appl.* **55**(12), 1–9 (2020)
9. Lyu, Q., et al.: Remotely access “my” smart home in private: an anti-tracking authentication and key agreement scheme. *IEEE Access* **7**, 41835–41851 (2019)
10. Alshahrani, M., Traore, I.F.: Secure mutual authentication and automated access control for IoT smart home using cumulative keyed-hash chain. *J. Inf. Secur. Appl.* **45**, 156–175 (2019)
11. Shuai, M., et al.: Anonymous authentication scheme for smart home environment with provable security. *Comput. Secur.* **86**, 132–146 (2019)
12. Banerjee, S., et al.: An efficient, anonymous and robust authentication scheme for smart home environments. *Sensors* **20**(4), 1215 (2020)
13. Xiaoying, J., et al.: A provably secure and efficient identity-based anonymous authentication scheme for mobile edge computing. *IEEE Syst. J.* **14**(1), 560–571 (2020)
14. Yuting, L., et al.: A secure anonymous identity based scheme in new authentication architecture for mobile edge computing. *IEEE Syst. J.* **15**(1), 560–571 (2020)
15. Burrows, M., Abadi, M., Needham, R.: A logic of authentication. *ACM Trans.* **8**(1), 18–36 (1990)
16. Armando, A., et al.: The AVISPA tool for the automated validation of internet security protocols and applications. In: Etessami, K., Rajamani, S.K. (eds.) *CAV 2005*. LNCS, vol. 3576, pp. 281–285. Springer, Heidelberg (2005). https://doi.org/10.1007/11513988_27



End-Edge Cooperative Scheduling Strategy Based on Software-Defined Networks

Fan Li, Ying Qiao, Juan Luo^(✉), Luxiu Yin, Xuan Liu, and Xin Fan

College of Computer Science and Electronic Engineering, Hunan University,
Changsha, China

{lifan118,qy2020,juanluo,yinluxiu,xuan_liu,xinfan}@hnu.edu.cn

Abstract. With the development of the Internet of Things (IoT), more and more applications are increasingly demanding latency. Traditional single-task scheduling strategy is difficult to satisfy low-latency demand. This is because the task scheduler usually schedules tasks to a closer server, which leads to an increase in task latency when there are more tasks, which in turn leads to an increase in task rejection rate. In this paper, we propose an end-edge cooperative multi-tasks scheduling (MTS) strategy based on improved particle swarm optimization (IPSO) algorithm. At first, we design a Software-Defined Networks controller algorithm to cluster task offload requests. Then, we set the scheduling priority for the multi-task clusters. At last, we minimize the total offloading cost of total tasks as the optimization goal to satisfy its delay. The results demonstrate that the strategy we proposed can effectively reduce the service cost of the system, and the processing delay of tasks, which improves the success rate of task processing.

Keywords: Edge computing · Resource management · Task offloading

1 Introduction

With the advent of 5G, the Internet has been promoted from the IoT to the Internet of Everything. Every walk of life is actively embracing changes. The world has set off a wave of digitization, and the data generated by terminal devices (TDs) is explosive growth. According to Cisco's estimation [1], by 2023, there will be 29.3 billion TDs (e.g., monitoring equipment, sensors, Robotics, and so on) in the world. Those TDs will connect Internet. Half of these connections will support a wide range of IoT applications (14.7 billion by 2023). As shown in Fig. 1, the wireless edge computing scenario is shown, many new applications such as AR, face recognition, 3D game video, autopilot, etc., which have higher delay requirements for time delay and security, have rapidly entered people's life. This has brought great challenges to cloud computing [2]. Edge computing [3] is proposed to solve the problem. Edge computing brings the computation ability from the cloud to the network of edge, which decreases the pressure of the bandwidth of the cloud. The computation-intensive applications can be offloaded to

the edge server. It allocates computing tasks to the edge for processing, alleviating the pressure of bandwidth and cloud data center, solving the problem that clouds computing cannot meet the low-latency task processing request.

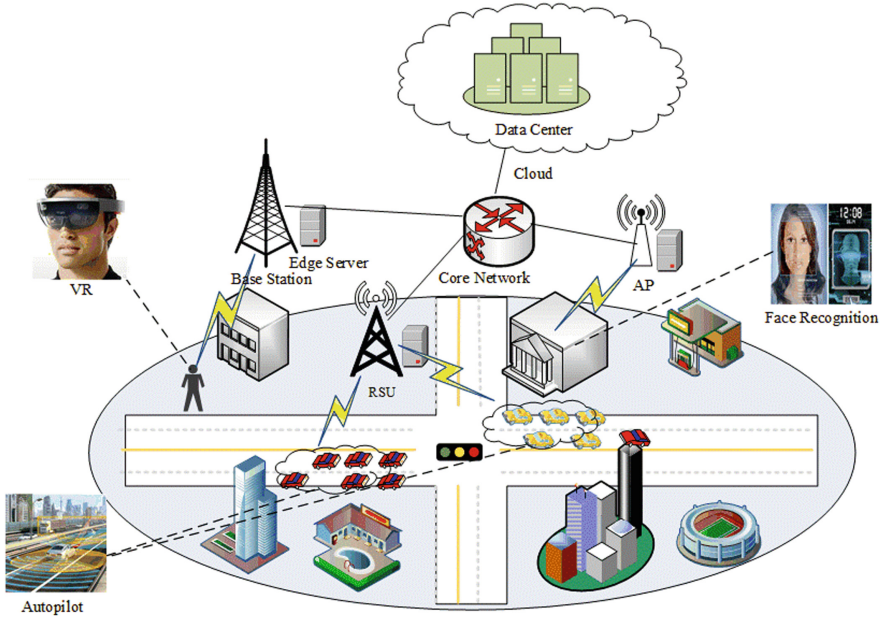


Fig. 1. The wireless edge computing scenario.

At present, many studies have shown that offloading all tasks to the edge servers still causes a large latency [4–8]. Many researchers have studied delay-sensitive tasks. Zhang et al. [9] proposed a task scheduling strategy for delay-sensitive tasks. They proved that the task scheduling problem is an NP-hard problem. However, the authors didn’t consider the problem of end-edge collaborative scheduling, and the model assumes that the next batch of tasks will be scheduled only after the current batch of tasks is processed. In [10], Yin et al. proposed a task scheduling algorithm based on Docker in fog computing, which was suitable for delay-sensitive and high concurrency. The author assumed that tasks were offloaded to the fog and cloud without reservation, and only considered the interaction between multiple TDs and a fog server. Intharawijitr et al. [11] proposed a fog computing model based on a 5G cellular network, which considered the application scenario of multi-terminal and multi-fog node interaction and took the goal of minimizing the task rejection rate. Through mathematical modeling of calculation and communication delay, a task scheduling algorithm with the lowest latency is proposed. The algorithm assigned each request to the fog node that can process the task with the lowest latency. However, the situation of local processing tasks is still not considered, and only takes the delay as

the decision-making reference factor, and does not consider the processing task cost and other factors.

As a new networking paradigm, software-defined networking (SDN) [12] can achieve logically centralized control on the distributed edge server and TDs, which can improve scheduling efficiency. However, facing the massive application market, how to meet the performance requirements of different applications is a challenge for edge computing. The essence of meeting the performance requirements of multiple applications is to complete the task request within the minimum required delay. At the same time, the development of the application market has led to the improvement of the performance of the terminal hardware. At present, most of the TDs on the market have certain computing performance. However, there are few types of research on end-edge cooperative scheduling of tasks. Most researchers consider offloading all tasks to the edge layer or cloud layer for processing, without considering the computing resources of TDs. There are two challenges in edge computing scheduling: (1) The TDs can only obtain the information of the request server, and it is difficult to obtain the information of global information, which leads to the low efficiency of offloading. (2) In the multi-task scenario, the system cost increases, and the single task scheduling strategy is difficult to meet the requirements of task delay, which leads to the failure of task offloading. To solve these problems, we propose an end-edge cooperative multi-task scheduling strategy based on IPSO.

Our contributions to this paper are as follows:

- Global information is not transparent in edge computing local networks, and intensive task scheduling is difficult to achieve optimality, resulting in high task rejection rates. We design an SDN controller based on edge application service architecture. SDN controller has a global network view that can flexibly manage edge servers and TDs. Therefore, this architecture can effectively improve the resource utilization of the edge layer and improve the quality of service of the TDs.
- The scheduler schedules tasks one after the other in a way that causes increased queuing delays for tasks and makes it difficult for the server to meet the latency requirements of the tasks. When the tasks of TDs near the server are more intensive, it leads to a full server task queue and an increased rejection rate. We present a multi-task joint scheduling strategy to minimize the total offloading cost of multiple tasks, to meet the minimum required delay of multiple tasks. We cluster the task offloading requests that arrive within a period before the task scheduling. Using the joint scheduling strategy, TDs can reduce the failure rate of task offloading.
- In MTS, we use particle swarm variational methods to improve the performance of scheduling. We improve the particle swarm optimization algorithm to solve the local optimum of traditional particle swarm optimization. When the particle swarm falls into the local optimum, we evaluate the particle positions and randomly initialize the particle positions with probability β , which improves the efficiency of task scheduling.

2 System Model

The architecture of the edge computing system based on SDN is shown in Fig. 2. The TD interacts with the SDN controller and edge server through the cellular base station, WiFi, AP, and other network access devices. As shown in Fig. 2, the task is offloaded in the order of ① to ④. Firstly, the TD sends the metadata of the task to the scheduling module of the SDN controller. The format of metadata is fixed, including the size of the task data, the minimum required to delay, the computing power of the TD, the remaining length of the cache queue of TD, and other information. Secondly, the scheduling model of the SDN controller reads the task request metadata and determines the task scheduling scheme according to the real-time information of the edge layer. The SDN controller sends the task scheduling scheme to the TDs. Thirdly, the TD offloads the task based on the scheduling scheme. Lastly, the edge server processes the tasks and sends a request to the TD. The TD does not send task to the SDN controller but sends the metadata of the task to the scheduling module of the SDN controller.

In this paper, we define the TDs with the set denoted as $\{T_1, T_2, T_3, \dots, T_N\}$, the edge server with the set denoted as $\{E_1, E_2, E_3, \dots, E_M\}$. To make it simple, we assume that the computation resource of each TD is a constant in each time slot denoted by u_t , and the computation resource of each edge server is u_e . We assume that the task requests generated by the TDs follow the Poisson distribution [11, 13].

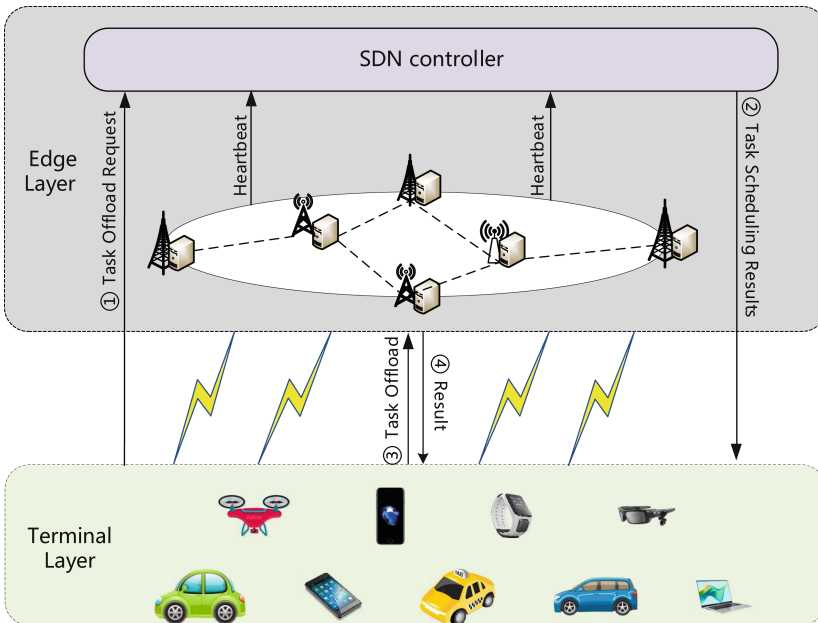


Fig. 2. Architecture of edge computing system based on SDN.

2.1 Task Cost

We combine the processing delay of the task and the cost of renting resources to define the system cost. The delay cost of $task_i$ mainly includes the transmission delay, queuing delay of the task, and the computing delay of the TD or the computing delay of the j -th server. The cost of renting resources mainly includes the cost of renting bandwidth resources and the cost of computing resources. The rental cost of the TD consists of the cost of renting bandwidth resources and renting computation resources of the edge server. We assume that the operator allocates a fixed bandwidth to the TDs, so we ignore the cost of renting bandwidth resources of TDs. If the cost of computing resources consumed by the edge server per unit time is c , The cost of renting computation resources is denoted as $Cost = C_{ij} * c$, where c is the cost of computing resources consumed by the edge server per unit time, C_{ij} is the computation delay of task i at edge server j .

2.2 System Delay

The system delay consists of the delay between the TD and the SDN controller, and the delay between the TD and the edge server. The delay between the TD and the SDN includes the transmission delay, the average queuing delay in the SDN controller, and the execution delay of the SDN controller. The TDs firstly send the metadata to the SDN controller, we define the metadata format as $Meta_i(data_i, delay_i, T_n^{cap})$, $data_i$ is the amount of task data, $delay_i$ is the minimum required delay, and T_n^{cap} represents the amount of tasks remaining in the queue of T_n . Due to the fixed format of task metadata and decision result data, little data is transmitted, and in the system architecture of Fig. 2, these two processes are essential regardless of the offloading result, so the delay of task scheduling results and execution result can be ignored. Therefore, we only consider the transmission delay of task data from the TD to the edge server. We assume that h_i and B are the channel gains and channel bandwidths of the TDs and the edge server. Let p_i denote the transmission power of the user, and σ^2 denotes the transmission noise. In this paper, the free space channel model is applied. Thus, the uplink data delay is given by

$$Tt_i = \frac{data_i}{B * \log_2 \left(1 + \frac{p_i * h_i}{\sigma^2} \right)}. \quad (1)$$

The SDN controller is set to wait for the number of offloading requests in the cache queue to reach a threshold δ before triggering the task scheduling algorithm. The TD generates task requests according to Poisson's expectations. If the number of TDs is N , the SDN controller's task arrival per unit time is $U = \lambda * N$, and the average queuing delay of processing requests is $Qs_i = \frac{1}{U} * \frac{\delta - 1}{2}$. If $task_i$ is offloaded to the TD for execution, then it has $Qt_i = \frac{T_n^{cap}}{u_t} + T_n^{cur}$, T_n^{cap} is the space occupied by the buffer queue of the TDs, and T_n^{cur} is the remaining execution time of the task being executed. If the SDN controller decides to offload the task to the j -th server for execution, the TD will send the task data to the

cache queue of the j -th server through the network access facility. The queue delays Qe_{ij} in the j -th server is equal to the sum of the total calculation time of the task to be processed in the cache queue of the current j -th server and the remaining execution time of the currently executing task expressed as:

$$Qe_{ij} = \frac{E_j^{cap}}{u_e} + E_j^{cur}, \tag{2}$$

and E_j^{cap} is the total amount of task data to be processed in the cache queue of the j -th server, and E_j^{cur} is the remaining execution time of the task being executed. The computing delay is the ratio of the data volume of the task to the data volume processed by the computing device per unit of time. Therefore, the computing delays of $task_i$ at the TD and j -th server are respectively expressed as $Ct_i = data_i/u_t$ and $C_{ij} = data_i/u_e$.

2.3 Problem Formulation

In this paper, we model the multi-task joint scheduling model of TDs and edge servers. Suppose that the edge server schedules δ task offloading requests $task_1, task_2, \dots, task_\delta$, edge layer has M available edge servers. We define the decision parameter x_{ij} , when $task_i$ has the least cost on edge server j $x_{ij} = 1$, else $x_{ij} = 0$.

The total delay of task offload requests in TD processing can be expressed as:

$$T^t = \sum_{i=1}^{\delta} \left[\left(1 - \sum_{j=1}^M x_{ij} \right) (Qs_i + Qt_i + Ct_i) \right] \tag{3}$$

where $\sum_{j=1}^M x_{ij} = 0$ means that the task is offloaded to the TD, Qs_i is SDN controller queuing delay, Ct_i is TD local computing time delay. The total delay cost of δ tasks on the edge server can be written as:

$$T^e = \sum_{i=1}^{\delta} \sum_{j=1}^M [x_{ij}(Tt_i + Qs_i + Qe_{ij} + C_{ij})]. \tag{4}$$

We aim to optimize the total offloading cost of multiple tasks under the constraint of the minimum required delay of each task. The optimization objective of total offloading cost can be summarized as follows:

$$\min \quad Cost_\delta \tag{5}$$

$$s.t. \quad \left(1 - \sum_{j=1}^M x_{ij} \right) (Qs_i + Ct_i) + \sum_{j=1}^M [x_{ij}(Tt_i + Qs_i + Qe_{ij} + C_{ij})] \leq delay_i, \tag{6}$$

where $Cost_\delta = T^t + T^e + \sum_{i=1}^{\delta} \sum_{j=1}^M x_{ij} C_{ij} * c$, which means total offloading cost of δ tasks, and its value is the sum of delay cost and rental resource cost. $delay_i$ means that the task scheduling needs to meet the minimum delay.

3 Collaborative End-Edge Multi-task Scheduling Strategy Based on Improved Particle Swarm

In this section, we propose an end-edge cooperative MTS strategy based on IPSO. The MTS strategy clusters the tasks and sets the scheduling priority according to the delay requirements of the tasks, which can ensure the first process of the delay-sensitive tasks. Because scheduling multiple tasks lead to a large space for problem-solving, the large scheduling delay increases the rejection rate of tasks and decreases the quality of experience of the TD. Therefore, we adopted the particle swarm optimization algorithm in the heuristic algorithm. To improve the searching ability of the algorithm in the solution space and the evolution speed of particles, we use the IPSO algorithm.

3.1 Task Classification

To better reflect the task delay sensitivity, we take the minimum required delay of the task and the amount of task calculation as the characteristic attributes of the task. The way we assess the degree of similarity between data is to calculate the Euclidean distance between task feature attributes. We first normalize the task's delay sensitivity and task volume. Then, we calculate the Euclidean distance of each task to the cluster center to complete the clustering of the task. We assume that the SDN controller caches the first δ tasks in the queue at time t , and the computational complexity and minimum required delay of these tasks are $\{data_1, data_2, \dots, data_\delta\}$ and $\{delay_1, delay_2, \dots, delay_\delta\}$. The task calculation amount is $\{data_1^*, data_2^*, \dots, data_\delta^*\}$, we have $data_i^* = \frac{data_i}{\sum_{j=1}^{\delta} data_j}$. The normalized value of the minimum required delay of the task is $\{delay_1^*, delay_2^*, \dots, delay_\delta^*\}$, where $delay_i^* = \frac{delay_i}{\sum_{j=1}^{\delta} delay_j}$. The similarity between $task_i$ and $task_j$ is $d_{ij} = \sqrt{(data_i^* - data_j^*)^2 + (delay_i^* - delay_j^*)^2}$. After clustering δ tasks, K task clusters will be generated.

3.2 Improved Particle Swarm Optimization Algorithm

According to the minimum delay requirement of K cluster centers, we set the scheduling priority for task clusters. The smaller the delay requirement is, the higher the priority is. In the task cluster scheduling stage, the particle swarm optimization algorithm has better optimization performance when using real coding, and this problem model just needs real coding to solve the MTS problem, so this paper uses the particle swarm optimization algorithm to schedule the task cluster. Each particle will remember its historical optimal position and population optimal position, and determine the direction and speed of the next step according to these two positions. Finally, the whole population will gather the optimal solution nearby, and the algorithm will take the optimal position of the population as the final solution to the problem. In this paper, each particle represents a scheduling scheme of the decision layer.

Particle coding methods are generally binary coding and real coding. Our coding method is integer coding in real coding. The dimension of particles is the number of tasks to be scheduled. Suppose that there are $S * N$ particles in the population, where par_s is the s -th particle in the population, and the position vector of the particle at time t is $Pos_s^t = (pos_{s,1}^t, pos_{s,2}^t, pos_{s,3}^t, \dots, pos_{s,\delta}^t)$, $pos_{s,i}^t \in [0, M]$, $pos_{s,i}^t = 0$ implies that SDN controller assign task i to the TD for local execution. $pos_{s,i}^t = j(j \in [1, M])$ means that TD will offload the task to the j -th server. The fitness function is used to evaluate the advantages and disadvantages of the particle position. Since the algorithm takes minimizing the system cost as the optimization goal, the smaller the fitness value of the particle, the higher the fitness value. The fitness function we use is based on the offload cost calculation (9). We use x_{ij} as the decision parameter, and $x_{ij} = 1$ indicates that the $task_i$ is offload to the j -th server. Therefore, we need to convert the particles into the formula. When the specific conversion scheme is $pos_{s,i}^t = j(j \neq 0)$, $x_{ij} = 1$, otherwise $x_{ij} = 0$.

In the iteration process, each particle will remember its optimal historical position $HPos_s^t$, and the global optimal position $GPos^t$ will be shared among particles. The particle position update is given by

$$\begin{cases} V_s^t = wV_s^{t-1} + c_1Rand_1 (HPos_s^t - Pos_s^{t-1}) + c_2Rand_2 (GPos^t - Pos_s^{t-1}) \\ Pos_s^t = Pos_s^{t-1} + V_s^t \end{cases}, \tag{7}$$

where c_1 and c_2 are learning factors, $Rand_1$ and $Rand_2$ are random number of (0,1) interval, and $Rand_1 + Rand_2 = 1$. Moreover, the particles have a chance of variation in the process of evolution, which means regenerating the random position, to improve the global search ability of the algorithm. w is the inertia factor, and its value is related to the global search ability. The larger the value is, the stronger the global search ability will be, while the local search ability will be weakened accordingly. The smaller the value is, the opposite is true. In the traditional particle swarm optimization algorithm, w is often set as a fixed value of 0.5, and this setting will lead to premature convergence in the early stage of the algorithm, and the population will fall into the local optimal solution early. Therefore, we use the linear decreasing weight strategy to dynamically modify w to improve the fine-grained search solution space in the early stage of the algorithm, and enhance the global search ability in the late stage of the algorithm, to improve the probability of the algorithm to obtain the optimal solution of the problem. The modified formula of w is

$$w = (w_{ini} - w_{end})(G - g)/G + w_{end}, \tag{8}$$

where w_{ini} is the initial value, w_{end} is the maximum number of iterations, G is the maximum number of iterations, and g is the current number of iterations. Due to the large probability of particle swarm optimization will fall into the local optimal solution, the problem of premature convergence appears. Therefore, a mechanism must be provided to make the algorithm jump out of the local optimum and continue to search in other regions of the solution space when

premature convergence occurs. In this paper, the mutation mechanism is used, that is, after evaluating the particle position, the probability β is used to initialize the particle position randomly.

We describe the pseudo-code of the proposed IPSO in Algorithm 1.

Algorithm 1: IPSO Algorithm

input : task data set $\{task_1, \dots, task_\delta\}$ the maximum number of iterations n
in k-means, the maximum number of iterations G

output: the global optimal particle position is the optimal scheduling result

- 1 *Initialize the cluster center and randomly select k tasks from δ to generate the cluster center.*
- 2 **for** $epoch = 1$ **to** n **do**
- 3 **for** $i = 1$ **to** δ **do**
- 4 **for** $j = 1$ **to** K **do**
- 5 $l_j = argmin_i d_{i,j}$
- 6 **end**
- 7 **end**
- 8 **for** $v = 1$ **to** K **do**
- 9 l_v center takes the average value of task quantity and delay of its category.
- 10 **end**
- 11 **end**
- 12 *The initial values of position and velocity are generated randomly.*
- 13 *The optimal particle is selected by fitness function and eq.6.*
- 14 **for** $g = 1$ **to** G **do**
- 15 *Change the inertia factor ω according to eq.8.*
- 16 *The particle swarm velocity and position are updated according to eq.7 update global optimum.*
- 17 β probability initializes particle position
- 18 **end**

4 Simulation and Evaluation

This section will evaluate the simulation performance of the multi-task joint scheduling strategy. We use the Eclipse simulation platform and the Java multi-threaded programming technology, which is to simulate the entire offloading process of tasks in edge computing.

4.1 Parameter Setting

In the simulation process, we take the image processing scene as the prototype and sets the parameters in the simulation delay. According to Poisson's expectation $\lambda = 1.5$, the TD generates 200 task processing requests. Since the size of

each picture is generally between 100 KB and 500 KB, the calculation amount of tasks sent by the TDs is set between 100 KB and 500 KB, and the minimum required delay of tasks is randomly generated within 1 s–5 s. In [14], the author presents that the computing power of the edge server is about 100–10K times that of the TDs, and the cache capacity is 100 times that of the TDs. Therefore, we set the configuration specification of the TDs which can calculate 500 KB of data per unit time, and cache queue capacity is 5 MB. The configuration specification of the edge server is the same, which can process 50 MB of data per unit time and cache 500 MB of queue capacity. For simplicity, we refer to [15] and assume the following simulation settings. Each device can access the access point through the communication channel. For each channel, the bandwidth B is 20 MHz, the white noise power σ^2 is -100 dBm, the channel model is independent of Rayleigh fading, the average power loss is 53 dB, and the transmission power of offloading is 500 mW. The SDN controller can schedule 500 task offloading requests per unit time. The cost of computing resources per unit time c is set to 80. In the MTS model, the task scheduling threshold δ is set to 30, and the number of task clusters K in the k-means algorithm is set to 4. In the IPSO, the number of populations is set to 20, the maximum number of iterations is $G = 30$, $c_1 = c_2 = 0.5$, and the number of times is $w_{init} = 0.4$, $w_{end} = 0.9$ and $\beta = 0.2$.

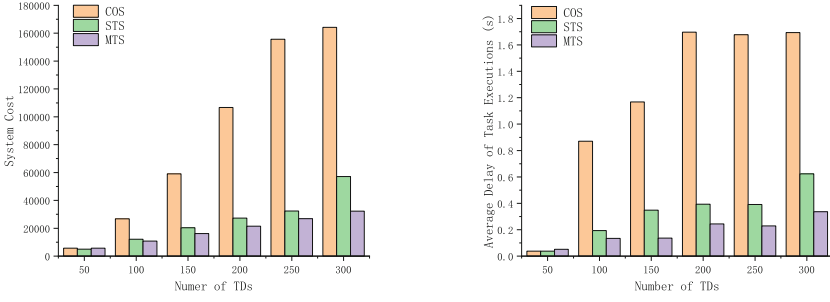
4.2 Simulation and Analysis

We compare MTS with two baseline methods, single task scheduling (STS) and complete offloading scheduling (COS).

- STS: A scheduling algorithm that offloads all tasks to the edge layer through the SDN controller using the greedy algorithm.
- COS: The strategy of offloading all tasks to the edge layer.

The experimental data of all comparison indexes are the average of 3 experimental results.

Impact of Computation Amount on Offloading Performance. At first, we consider the impact of the number of TDs on offloading performance. At this time, the number of edge servers is set to 20. As shown in Fig. 3, with the increase of TDs, the system cost and task rejection rate increase. From Fig. 3(a), we can see that the system cost of MTS is higher than the other two algorithms at the beginning, and the system cost is minimized as the tasks increase. Then, we can conclude that MTS requires tasks to reach the threshold before scheduling, and the algorithm complexity is higher than the other two strategies, resulting in large task scheduling delays. In Fig. 3(b), we can see that the average delay increases with the increase of tasks. When the number of tasks increases to a certain level, the COS delay does not change significantly, as is shown in Fig. 3(b). This is because the edge node can't meet the minimum delay required to execute the task, and the task request is rejected.



(a) Influence of TD number on the system cost (b) Influence of TD number on average task delay

Fig. 3. The impact of the number of TDs on the task.

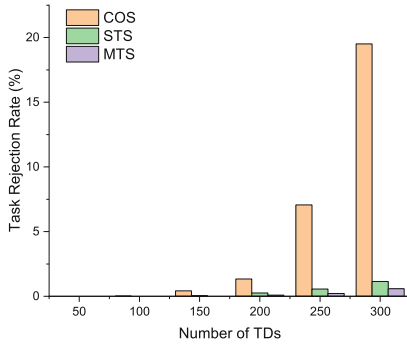
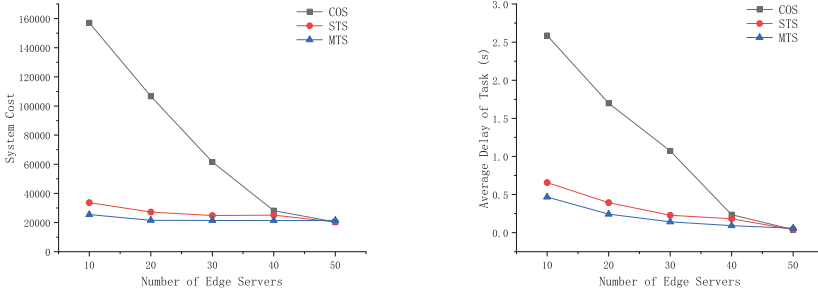


Fig. 4. The influence of TD number on task rejection rate.

As shown in Fig. 4, with the increase in the number of TDs, the rejection rate of COS even reached 19.5 %, which also verified the above statement. The rejection rate of MTS is nearly half lower than that of STS. We can conclude that our algorithm MTS has a lower task rejection rate than other algorithms, indicating that our algorithm has a good effect on improving the quality of service.

Impact of Computation Amount on Offloading Performance. The impact of the number of edge servers on the system cost is shown in Fig. 5(a). In the process of increasing the number of edge servers, gradually abundant computing resources reduce the processing delay of tasks, and the system cost of the three strategies will also decrease correspondingly. When the number of edge servers exceeds 30, the system cost tends to be stable State. When the number of edge servers increases to 50, the extremely low processing delay of the edge layer causes the tasks to be offloaded to the edge layer. From the experimental results, it can be concluded that the total cost of the system can't be reduced by increasing the number of edge servers. When the computing resources of the edge layer are sufficient, the system cost tends to be stable. We also analyze the influ-



(a) The impact of the number of edge servers on system cost (b) The influence of the number of edge servers on the average task delay

Fig. 5. The impact of the number of edge servers on tasks.

ence of the number of edge servers on the total task delay, and the experimental results are shown in Fig. 5(b). It can be seen from the figure that the strategies we proposed can effectively reduce the processing delay of tasks and improve the service experience of users in the situation of a shortage of computing resources or abundant computing resources.

5 Conclusion

In this paper, we design an end-edge cooperative MTS strategy with the SDN controller’s global function. The system aims to minimize the total offloading cost of multiple tasks under the constraint of the minimum required delay of each task. We use the IPSO algorithm to solve the optimization problem. Simulation results show that the scheduling strategies can meet the TD delay requirements, reduce the average processing delay of tasks, improve resource utilization, and reduce the system cost. The system architecture of this paper only considers the task scheduling in one region. Our next work will consider the joint task scheduling of SDN controllers in multiple regions. Through the mutual communication between multiple regional SDN controllers, and sharing the resource status information of the edge layer, we can build a more complete network view, provide a more perfect scheduling scheme for the TD layer, expand the service scope of the edge layer resources, and improve the resource utilization of the edge layer.

Acknowledgment. This work was supported in part by the National Natural Science Foundation of China under Grant 61972140 and 62002109, and the National Defense Basic Research Plan under Grant JCKY2018110C145.

References

1. Cisco, U.: Cisco annual internet report (2018–2023) White paper (2020)
2. Armbrust, M., et al.: A view of cloud computing. *Commun. ACM* **53**(4), 50–58 (2010)
3. Shi, W., Cao, J., Zhang, Q., Li, Y., Xu, L.: Edge computing: vision and challenges. *IEEE Internet Things J.* **3**(5), 637–646 (2016). <https://doi.org/10.1109/JIOT.2016.2579198>
4. Tuli, S., Ilager, S., Ramamohanarao, K., Buyya, R.: Dynamic scheduling for stochastic edge-cloud computing environments using A3C learning and residual recurrent neural networks. In: *IEEE Transactions on Mobile Computing/* <https://doi.org/10.1109/TMC.2020.3017079>
5. Yang, L., Cao, J., Cheng, H., Ji, Y.: Multi-user computation partitioning for latency sensitive mobile cloud applications. *IEEE Trans. Comput.* **64**(8), 2253–2266 (2015). <https://doi.org/10.1109/TC.2014.2366735>
6. Uргаonkar, R., et al.: Dynamic service migration and workload scheduling in edge-clouds. *Perform. Eval.* **91**, 205–228 (2015)
7. Zeng, D., Gu, L., Guo, S., Cheng, Z., Yu, S.: Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system. *IEEE Trans. Comput.* **65**(12), 3702–3712 (2016). <https://doi.org/10.1109/TC.2016.2536019>
8. Miranda, C., Kaddoum, G., Baek, J.-Y., Selim, B.: Task allocation framework for soft-ware-defined fog v-RAN. *IEEE Internet Things J.* <https://doi.org/10.1109/JIOT.2021.3068878>
9. Zhang, Y., Chen, X., Chen, Y., Li, Z., Huang, J.: Cost efficient scheduling for delay-sensitive tasks in edge computing system. In: 2018 IEEE International Conference on Ser-vices Computing (SCC), San Francisco, CA, pp. 73–80 (2018)
10. Yin, L., Luo, J., Luo, H.: Tasks scheduling and resource allocation in fog computing based on containers for smart manufacturing. *IEEE Trans. Ind. Inform.* **14**(10), 4712–4721 (2018). <https://doi.org/10.1109/TII.2018.2851241>
11. Intharawijitr, Iida, K., Koga, H.: Analysis of fog model considering computing and communication latency in 5G cellular networks. In: 2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops), Sydney, NSW, Australia, pp. 1–4 (2016). <https://doi.org/10.1109/PERCOMW.2016.7457059>
12. Chen, M., Hao, Y.: Task offloading for mobile edge computing in software defined ultra-dense network. *IEEE J. Sel. Areas Commun.* **36**(3), 587–597 (2018). <https://doi.org/10.1109/JSAC.2018.2815360>
13. Chang, Z., Liu, L., Guo, X., Sheng, Q.: Dynamic resource allocation and computation offloading for iot fog computing system. *IEEE Trans. Ind. Inform.* **17**(5), 3348–3357 (2021). <https://doi.org/10.1109/TII.2020.2978946>
14. Cuervo E., Balasubramanian A., Cho D.K., et al.: MAUI: making smartphones last longer with code offload[. In: *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services (MobiSys 2010)*, San Francisco, California, USA, 15–18 June 2010. DBLP (2010)
15. Yang, X., Luo, H., Sun, Y., Zou, J., Guizani, M.: Coalitional game based cooperative computation offloading in MEC for reusable tasks. *IEEE Internet Things Journal.* <https://doi.org/10.1109/JIOT.2021.3064186>



Joint Optimization of Bandwidth Allocation and Gradient Quantization for Federated Edge Learning

Hao Yan^{1,2}, Bin Tang^{1,2(✉)}, and Baoliu Ye^{1,2}

¹ Key Laboratory of Water Big Data Technology of Ministry of Water Resources, Hohai University, Nanjing 211100, China

² School of Computer and Information, Hohai University, Nanjing 211100, China
{yanhao, cstb, yeb1}@hhu.edu.cn

Abstract. Federated Edge Learning (FEEL) is becoming a popular distributed privacy-preserving machine learning (ML) framework where multiple edge devices collaboratively train an ML model with the help of an edge server. However, FEEL usually suffers from a communication bottleneck due to the limited sharing wireless spectrum as well as the large size of training parameters. In this paper, we consider gradient quantization to reduce the communication traffic and aim at minimizing the total training latency. Since the per-round latency is determined by both the bandwidth allocation scheme and gradient quantization scheme (i.e., the quantization levels of edge devices), while the number of training rounds is affected by the latter, we propose a joint optimization of bandwidth allocation and gradient quantization. Based on the analysis of total training latency, we first formulate the joint optimization problem as nonlinear integer programming. To solve this problem, we then consider a variation of this problem where the per-round latency is fixed. Although this variation is proved to be NP-hard, we show that it can be transformed into a multiple-choice knapsack problem which can be solved efficiently by a pseudopolynomial time algorithm based on dynamic programming. We further propose a ternary search based algorithm to find a near-optimal per-round latency, so that the two algorithms together can yield a near-optimal solution to the joint optimization problem. The effectiveness of our proposed approach is validated through simulation experiments.

Keywords: Federated edge learning · Gradient quantization · Bandwidth allocation

1 Introduction

Federated Edge Learning (FEEL) [8, 11, 13] is becoming a popular distributed privacy-preserving machine learning (ML) framework where multiple edge devices collaboratively train an ML model with the help of an edge server. In FEEL, edge devices compute gradients for a global model on their local data,

and upload the gradients to the edge server for model updates in an iterative manner. However, FEEL usually suffers severely from the communication bottleneck caused by the limited sharing wireless spectrum as well as the large size of training parameters. One promising approach to alleviate the communication bottleneck is gradient quantization, which refers to using fewer bits to represent gradients. For example, [3, 12] use 1-bit quantization to increase the speed of model training with little loss of accuracy. TernGrad [15] maps positive, zero, and negative elements in gradients into 1, 0 and -1. QSGD [1] provides users with different quantization levels to choose from, thus making a trade-off between communication cost and convergence speed.

In this paper, we consider gradient quantization to reduce the communication traffic and aim at minimizing the total training latency, which is a critical concern in FEEL [10]. The total training latency is proportional to the number of required training rounds and the per-round latency. The former is closely related to the gradient quantization scheme, i.e., the individual quantization levels of edge devices. The latter is determined by the gradient quantization scheme together with the bandwidth allocation scheme which specifies how the spectrum is shared among the edge devices. Bandwidth allocation and gradient quantization are closely related, e.g., when an edge device uses a higher quantization level which leads to larger communication traffic, it requires more bandwidth to shorten its transmission time. On the other hand, the per-round latency is determined by the slowest edge device. Hence, it is necessary to optimize bandwidth allocation and gradient quantization for all the edge devices in a joint manner.

Several works [4, 6, 9, 10, 14] have considered the joint optimization of bandwidth allocation and gradient quantization in different contexts. More specifically, in [4, 6, 14], a stochastic quantization method proposed in [2] is considered, where the quantization level at each edge server relies on the dynamic range of its local gradient. So the joint optimization can only be done when all the edge devices have carried out their local gradients, which is unfavorable since the computation faster edge devices need to wait for slower devices before starting their transmissions. In contrast, [9, 10] consider another quantization method proposed in [1], where the quantization levels of edge servers are independent with the values of their local gradients. So the optimization can be done before each training round starts. However, in [9], only the number of training rounds is optimized but the per-round latency is not taken into account, while in [10], all the edge servers use the same quantization level, which only leads to an inferior performance as demonstrated by our experiments.

In this paper, we consider a generalization of the quantization method proposed in [1], where the edge servers can adopt different quantization levels, and minimize the total training latency by optimizing the bandwidth allocation and gradient quantization in a joint manner. We first characterize the total training latency under given bandwidth allocation scheme and gradient quantization scheme, and then formulate the joint optimization problem as nonlinear integer programming. To solve this problem, we propose a variation of this problem where the per-round latency is fixed and treated as a constraint other than a part of objective function. Although this variation is proved to be NP-hard, we

show that this variation can be transformed into the well-known multiple-choice knapsack problem which can be solved efficiently by a pseudopolynomial time algorithm based on dynamic programming. In order to find a near-optimal per-round latency, we further propose a ternary search based algorithm. Combining with the algorithm for the variation, this can yield a near-optimal solution to the joint optimization problem. Finally, we demonstrate the effectiveness of our proposed approach through simulation experiments.

2 System Model

As shown in Fig. 1, we consider a federated edge learning (FEEL) system consisting of an edge server and a set of M edge devices. The edge devices connect to the edge server via a shared wireless spectrum. Each edge device m , $m = 1, \dots, M$, has a local data set \mathcal{D}_m which consists of $N_m = |\mathcal{D}_m|$ data samples. We assume that these \mathcal{D}_m are disjoint, and denote the whole dataset by $\mathcal{D} = \cup_{m=1}^M \mathcal{D}_m$ which has size $N = \sum_{m=1}^M N_m$. The aim of FEEL is to collaboratively train a machine learning model $\mathbf{w} \in \mathbb{R}^d$ that minimizes an empirical loss function,

$$L(\mathbf{w}) = \frac{1}{N} \sum_{m=1}^M \sum_{(\mathbf{x}, y) \in \mathcal{D}_m} f(\mathbf{w}, \mathbf{x}, y), \quad (1)$$

where $f(\mathbf{w}, \mathbf{x}, y)$ denotes the sample-wise loss function. The minimization is done by the system using gradient descent (GD) algorithm, which consists of multiple rounds. In the t -th round, edge server broadcasts global model \mathbf{w}^t to all users. Each edge device m computes its local gradient

$$\mathbf{g}_m(\mathbf{w}^t) = \frac{1}{N_m} \sum_{(\mathbf{x}, y) \in \mathcal{D}_m} \nabla f(\mathbf{w}^t, \mathbf{x}, y). \quad (2)$$

To reduce the communication traffic, it then uploads a quantized version of its local gradient, denoted by $Q_m(\mathbf{g}_m(\mathbf{w}^t))$ to the edge server. After receiving all the local gradients of the edge devices, the edge server updates the global model as

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \frac{\eta}{N} \sum_{m=1}^M N_m Q_m(\mathbf{g}_m(\mathbf{w}^t)), \quad (3)$$

where η denotes the learning rate.

A widely-used stochastic quantization method is considered for local gradient quantization [1]. The quantization function $Q_m : \mathbb{R}^d \rightarrow \mathbb{R}^d$ for edge device m is defined as follows. For any gradient $\mathbf{g} \in \mathbb{R}^d$ that is not a zero vector, and $1 \leq i \leq d$, the i -th entry of $Q_m(\mathbf{g})$ is $\|\mathbf{g}\|_2 \cdot \text{sgn}(g_i) \cdot \xi_i(\mathbf{g}, s_m)$, where g_i denotes the i -th entry of \mathbf{g} , $\|\mathbf{g}\|_2$ denotes the 2-norm of \mathbf{g} , s_m is a positive integer referred to as the quantization level of edge server m , and $\xi_i(\mathbf{g}, s_m)$ is an independent random variable defined as follows. Let ℓ be an integer such that

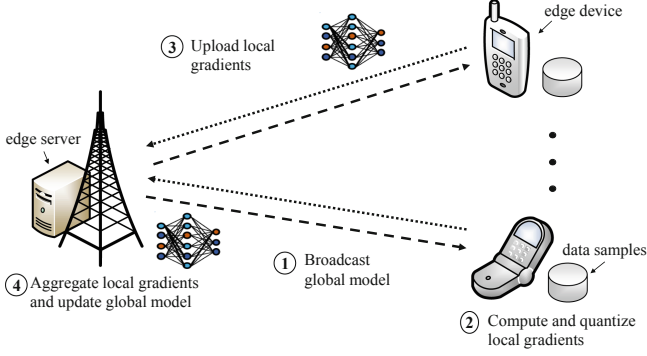


Fig. 1. An illustration of FEEL system.

$|g_i| / \|\mathbf{g}\|_2 \in [\ell/s_m, (\ell+1)/s_m]$, i.e., $[\ell/s_m, (\ell+1)/s_m]$ is the quantization interval corresponding to $|g_i| / \|\mathbf{g}\|_2$. Then

$$\xi_i(\mathbf{g}, s_m) = \begin{cases} (\ell+1)/s_m & \text{w.p. } \frac{|g_i|}{\|\mathbf{g}\|_2} s_m - \ell \\ \ell/s_m & \text{otherwise.} \end{cases} \quad (4)$$

If $\mathbf{g} = \mathbf{0}$, then we define $Q_m(\mathbf{g}, s) = \mathbf{0}$. As proved in [1], the quantization function satisfies the following properties: (1) Unbiasedness: $\mathbb{E}[Q_m(\mathbf{g})] = \mathbf{g}$. (2) Variance Upper Bound: $\text{Var}(Q_m(\mathbf{g})) = \mathbb{E}[\|Q_m(\mathbf{g}) - \mathbf{g}\|_2^2] \leq \min\left(\frac{d}{s_m^2}, \frac{\sqrt{d}}{s_m}\right) \|\mathbf{g}\|_2^2$.

3 Problem Formulation

We start by characterizing the total training latency, which depends on the number of training rounds required as well as per-round latency.

3.1 Number of Training Rounds

We first analyze the variance of the weighted average gradient aggregated by the edge server $\mathbf{g}^t = \frac{1}{N} \sum_{m=1}^M N_m Q_m(\mathbf{g}_m(\mathbf{w}^t))$. We have

$$\begin{aligned} \text{Var}(\mathbf{g}^t) &= \frac{1}{N^2} \sum_{m=1}^M N_m^2 \text{Var}(Q_m(\mathbf{g}_m(\mathbf{w}^t))) \\ &\leq \frac{1}{N^2} \sum_{m=1}^M N_m^2 \min\left(\frac{d}{s_m^2}, \frac{\sqrt{d}}{s_m}\right) \|\mathbf{g}_m(\mathbf{w}^t)\|_2^2 \\ &\leq \frac{1}{N^2} \sum_{m=1}^M N_m^2 \min\left(\frac{d}{s_m^2}, \frac{\sqrt{d}}{s_m}\right) \frac{1}{N_m^2} \sum_{(\mathbf{x}, y) \in \mathcal{D}_m} \|\nabla f(\mathbf{w}^t, \mathbf{x}, y)\|_2^2 \\ &\leq \left(\frac{1}{N^2} \sum_{m=1}^M \min\left(\frac{d}{s_m^2}, \frac{\sqrt{d}}{s_m}\right) N_m\right) Z \triangleq \sigma^2, \end{aligned}$$

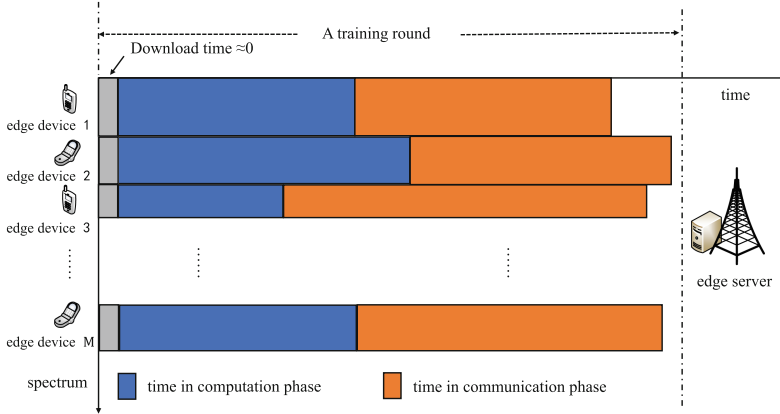


Fig. 2. Illustration of a training round.

where the last step holds under a widely used assumption that there exists some constant Z that satisfies $\|\nabla f(\mathbf{w}^t, \mathbf{x}, y)\|_2^2 \leq Z$ for any \mathbf{w}^t and $(\mathbf{x}, y) \in \mathcal{D}$. According to [5, Theorem 6.3], we have the following result directly.

Theorem 1. *Assume that the loss function $f(\cdot)$ is convex and ℓ -smooth. Let $\mathbf{w}_0 \in \mathbb{R}^d$ be given and $R^2 = \sup_{\mathbf{w} \in \mathbb{R}^d} \|\mathbf{w} - \mathbf{w}_0\|_2^2$. Then FEEL with initial model \mathbf{w}_0 and learning rate $\eta = \frac{1}{\ell+1/\gamma}$, where $\gamma = \frac{R}{\sigma} \sqrt{\frac{2}{T}}$, achieves*

$$\mathbb{E} \left[L \left(\frac{1}{T} \sum_{t=0}^T \mathbf{w}_t \right) \right] - \min_{\mathbf{w} \in \mathbb{R}^d} L(\mathbf{w}) \leq R \sqrt{\frac{2\sigma^2}{T}} + \frac{\ell R^2}{T}. \tag{5}$$

From Theorem 1, we can see that, in order to achieve an ϵ -optimality gap, i.e., $\mathbb{E} \left[L \left(\frac{1}{T} \sum_{t=0}^T \mathbf{w}_t \right) \right] - \min_{\mathbf{w} \in \mathbb{R}^d} L(\mathbf{w}) \leq \epsilon$, the number of required training rounds, denoted by $T(\epsilon)$, satisfies that $T(\epsilon) = O \left(R^2 \cdot \max \left(\frac{2\sigma^2}{\epsilon^2}, \frac{\ell}{\epsilon} \right) \right)$.

When the bandwidth is rather limited, the quantization levels s_m usually have to be small integers, and hence $\frac{2\sigma^2}{\epsilon^2}$ is larger than $\frac{\ell}{\epsilon}$ when ϵ is very small. In the following, we assume

$$T(\epsilon) = O \left(R^2 \cdot \frac{2\sigma^2}{\epsilon^2} \right) \tag{6}$$

3.2 Per-round Latency

As illustrated in Fig. 2, the process of a training round at each edge device consists of three phases: model downloading phase which is used for downloading the global model, the computation phase, and the communication phase. Usually, the time of the first phase is much smaller than the time of the other two phases,

so we assume that the downloading time is zero. For the computation phase, let ν denote the number of processing cycles required by each edge device to process one sample, and F_m denotes the clock frequency of edge device m , then the computation time of m is $t_m^{\text{cp}} = \frac{N_m \cdot \nu}{F_m}$.

For the communication phase, the communication time of each edge device depends on how the whole spectrum of frequency bandwidth B is shared among edge devices. In this paper, we consider a frequency-division multiple access (FDMA) is adopted for uplink transmission. Suppose each edge device m is allocated with a non-overlapping spectrum with frequency bandwidth b_m . According to Shannon's second theorem, the data transfer rate of node m is given as

$$r_m = b_m \log_2 \left(1 + \frac{P_m h_m^2}{b_m N_0} \right), \quad (7)$$

where P_m denotes the transmit power at edge device m , h_m denotes the channel propagation coefficient between m and the edge server, and N_0 denotes noise power spectral density.

For uploading the quantized local gradient $Q_m(\mathbf{g}_m(\mathbf{w}^t))$, edge device m needs to encode the vector norm $\|\mathbf{g}_m(\mathbf{w}^t)\|_2$, $\text{sgn}(\mathbf{g}_m(\mathbf{w}^t))$, as well as $Q_m(\mathbf{g}_m(\mathbf{w}^t))$ into bits. Encoding of $\text{sgn}(\mathbf{g}_m(\mathbf{w}^t))$ costs d bit bits, one bit for each entry. Encoding of $Q_m(\mathbf{g}_m(\mathbf{w}^t))$ costs $d \lceil \log_2(s_m + 1) \rceil$ bits, where each entry costs $\lceil \log_2(s_m + 1) \rceil$ bits. Encoding $\|\mathbf{g}_m(\mathbf{w}^t)\|_2$ costs a constant number of bits, which is negligible. Hence, the communication time of m is

$$t_m^{\text{cm}} = \frac{d(\lceil \log_2(s_m + 1) \rceil + 1)}{r_m}. \quad (8)$$

Since edge devices are usually in shortage of energy, we impose the following constraint on energy consumption for both computation and communication:

$$E_m^{\text{cp}} + P_m t_m^{\text{cm}} \leq E_m, \quad m = 1, 2, \dots, M \quad (9)$$

where E_m^{cp} is the energy cost for computation, and E_m is the maximum energy consumption of m per round. Finally, since the edge server needs to collect the local gradients of all edge devices before updating the model, the total latency per round is

$$t^{\text{round}} = \max_{1 \leq m \leq M} t_m^{\text{cp}} + t_m^{\text{cm}}. \quad (10)$$

3.3 Joint Bandwidth Allocation and Gradient Quantization

According to (6), the number of training rounds for achieving an ϵ -optimality gap is proportional to σ^2 . So in order to minimize the total training latency, one natural objective function is

$$\sigma^2 \cdot t^{\text{round}} = \left(\frac{1}{N^2} \sum_{m=1}^M \min \left(\frac{d}{s_m^2}, \frac{\sqrt{d}}{s_m} \right) N_m \right) Z \cdot t^{\text{round}} \quad (11)$$

Generally speaking, when the available spectrum is somehow inadequate or the available energy of edge devices is low, s_m should be set small. On the other hand, the dimension of model d is very large. So for simplicity, we assume that $\min\left(\frac{d}{s_m^2}, \frac{\sqrt{d}}{s_m}\right) = \frac{\sqrt{d}}{s_m}$. Then we can minimize the total training latency by minimizing $\sum_{m=1}^M \frac{N_m}{s_m} \cdot t^{\text{round}}$.

Hence, the joint optimization problem of bandwidth allocation and gradient quantization (BA-GQ), which is to determine $\{b_m\}$, the bandwidth allocation, and $\{s_m\}$, the quantization levels of edge devices, can be formulated as follows:

$$(P1): \quad \min \sum_{m=1}^M \frac{N_m}{s_m} \cdot t^{\text{round}} \quad (12)$$

$$\text{s.t.} \quad t_m^{\text{cp}} + \frac{d(\lceil \log_2(s_m + 1) \rceil + 1)}{b_m \log_2\left(1 + \frac{P_m h_m^2}{b_m N_0}\right)} \leq t^{\text{round}}, \forall m \quad (12a)$$

$$E_m^{\text{cp}} + P_m \frac{d(\lceil \log_2(s_m + 1) \rceil + 1)}{b_m \log_2\left(1 + \frac{P_m h_m^2}{b_m N_0}\right)} \leq E_m, \forall m \quad (12b)$$

$$\sum_{m=1}^M b_m \leq B \quad (12c)$$

$$s_m \in \mathbb{N}^+, s_m \leq 2^{\hat{q}} - 1, \forall m \quad (12d)$$

where (12b) corresponds to the energy constraint (9), (12c) is the bandwidth constraint, and (12d) means that the maximum quantization level is $2^{\hat{q}} - 1$ which corresponds to that the full-precision local gradient (i.e., without quantization) is represented by \hat{q} bits in edge devices. One common value of \hat{q} is 32.

Define $q_m = \lceil \log_2(s_m + 1) \rceil$. Then $s_m \leq 2^{q_m} - 1$. It is straightforward to see that, in any optimal solution of (P1), $s_m = 2^{q_m} - 1$. Therefore, we can transform (P1) into the following form:

$$(P2): \quad \min \sum_{m=1}^M \frac{N_m}{2^{q_m} - 1} \cdot t^{\text{round}} \quad (13)$$

$$\text{s.t.} \quad t_m^{\text{cp}} + \frac{d(q_m + 1)}{b_m \log_2\left(1 + \frac{P_m h_m^2}{b_m N_0}\right)} \leq t^{\text{round}}, \forall m \quad (13a)$$

$$E_m^{\text{cp}} + P_m \frac{d(q_m + 1)}{b_m \log_2\left(1 + \frac{P_m h_m^2}{b_m N_0}\right)} \leq E_m, \forall m \quad (13b)$$

$$q_m \in \mathbb{N}^+, 1 \leq q_m \leq \hat{q}, \forall m \quad (13c)$$

4 Algorithm Design

The BA-GQ problem is generally hard, as both the objective function (13) and the constraint (13a) are nonlinear. To tackle this issue, we first consider a

variation of BA-GQ where the per-round latency t^{round} is fixed to be a constant, and propose an efficient algorithm for this variation, and then show how to find an optimal t^{round} .

4.1 BA-GQ with Fixed Per-Round Latency

By fixing per-round latency t^{round} to be a constant, we have the following programming problem, which is referred to as BA-GQ-variation:

$$(P2): \quad \min_{b_m, q_m} \sum_{m=1}^M \frac{N_m}{2^{q_m} - 1} \quad (14)$$

$$\text{s.t.} \quad b_m \log_2 \left(1 + \frac{P_m h_m^2}{b_m N_0} \right) \geq \frac{d(q_m + 1)}{t^{\text{round}} - t_m^{\text{cp}}}, \forall m \quad (14a)$$

$$db_m \log_2 \left(1 + \frac{P_m h_m^2}{b_m N_0} \right) \geq \frac{P_m d(q_m + 1)}{E_m - E_m^{\text{cp}}}, \forall m \quad (14b)$$

$$\sum_{m=1}^M b_m \leq B \quad (15)$$

$$q_m \in \mathbb{N}^+, 1 \leq q_m \leq \hat{q}, \forall m \quad (16)$$

In the following, we will show that BA-GQ-variation is NP-hard, and further propose an efficient algorithm to solve it. Let $b_m^*(q_m)$ be the minimum value of b_m such that the corresponding constraints (14a) and (14b) are satisfied. Since $b_m \log_2 \left(1 + \frac{P_m h_m^2}{b_m N_0} \right)$ is a strictly increasing function of b_m , $b_m^*(q_m)$ exists and is unique. Hence, (P3) is equivalent to the following programming:

$$(P4): \quad \min \sum_{m=1}^M \frac{N_m}{2^{q_m} - 1} \quad (17)$$

$$\text{s.t.} \quad b_m \geq b_m^*(q_m) \quad (17a)$$

(12c), (13a), (13b)

Theorem 2. *The BA-GQ-variation problem given in (P3) is NP-hard.*

Proof. We will establish a polynomial-time reduction from the well-known 0-1 knapsack problem which is NP-hard. The 0-1 knapsack problem is defined as follows: given a set of M items, each with a weight w_m and a value v_m , along with a maximum weight capacity W , the objective is to find a subset of items such that the total weight of these items does not exceed W while the total value is maximized. We will reduce it to an instance of the BA-GQ-variation problem which is generated as follows: set $\hat{q} = 2$, and find proper values of input values such that $b_m^*(2) - b_m^*(1) = w_m$. Also $B = W + \sum_{m=1}^M b_m^*(1)$. Besides, $N_m = \frac{3}{2}v_m$.

We note that there is a one-one correspondence between the solutions of the instance of 0-1 knapsack problem and the solutions of the instance of the BA-GQ-variation problem, which is defined as follows: let $\mathcal{A} \subseteq \{1, 2, \dots, M\}$ be a feasible solution of the instance of 0-1 knapsack problem with objective value SOL , its corresponding feasible solution of the instance of the BA-GQ-variation problem is given as $q_m = 2$ and $b_m = b_m^*(2)$ for $m \in \mathcal{A}$, and $q_m = 1$ and $b_m = b_m^*(1)$ for $m \notin \mathcal{A}$, and vice versa. Besides, the solution of the instance of 0-1 knapsack problem has an objective value of $\sum_{m \in \mathcal{A}} \frac{1}{2}v_m + \sum_{m \notin \mathcal{A}} \frac{3}{2}v_m = \frac{3}{2} \sum_{m=1}^M v_m - SOL$. This implies that the instance of 0-1 knapsack problem has an optimal value of OPT if and only if the instance of the BA-GQ-variation problem is $\frac{3}{2} \sum_{m=1}^M v_m - OPT$. The proof is accomplished by noting that the reduction can be done in polynomial time.

From the proof of Theorem 2, we can see there exists some connection between the BA-GQ-variation problem and the knapsack problem. In fact, we can formulate the BA-GQ-variation problem as an equivalent maximization problem:

$$\begin{aligned}
 \text{(P5):} \quad & \max \sum_{m=1}^M \left(U - \frac{1}{2^{q_m} - 1} \right) N_m \\
 \text{s.t.} \quad & \text{(12c), (13a), (13b), (17a)}
 \end{aligned} \tag{18}$$

One key observation is that (P5) can be viewed as the multiple-choice knapsack problem (MCKP) [7], where the knapsack has a weight capacity of B , the set of items consists of M classes, each class \mathcal{C}_m , $m = 1, \dots, M$, consists of \hat{q} items, each having weight $b_m^*(q_m)$ and value $\left(U - \frac{1}{2^{q_m} - 1} \right) N_m$, $q_m = 1, \dots, \hat{q}$. U is a large constant to ensure the value is positive. The problem is to select exactly one item from each class such that the total weight does not exceed B while the total value is maximized. Despite the NP-hardness of MCKP, several efficient algorithms have been proposed to solve MCKP, including dynamic-programming based pseudo-polynomial time algorithm which can return an optimal solution, and polynomial-time approximation scheme (PTAS) that can produce a solution that is within a factor of $1 - \varepsilon$ of being optimal for any constant $\varepsilon > 0$ in polynomial time of the problem size [7]. In this paper, we employ the dynamic programming algorithm to solve (P5), which yields an optimal solution to (P3).

4.2 Search for Optimal Per-Round Latency

For a fixed t^{round} , let the optimal objective value of (P3) be $F(t^{\text{round}})$, which can be computed by Algorithm 1. Now we proceed to introduce how to find an optimal round time t^{round} such that $F(t^{\text{round}})t^{\text{round}}$ is optimized.

Consider a large enough value T_{up} which is an upper bound of t^{round} and $T_{low} = \max\{t_m^{\text{CP}}\}$ which is a lower bound, such that the search space is $[T_{low}, T_{up}]$. In general, $F(t^{\text{round}})t^{\text{round}}$ is not a function that is first decreasing then increasing of t^{round} , which is due to the integrity requirement of q_m . To address this, we discretize the search space $[T_{low}, T_{up}]$ into Ψ points with equal distance, where Ψ

is a tunable parameter. When Ψ is not very large, e.g., $\Psi = 100$, we observe from our experiments that $F(t^{\text{round}})t^{\text{round}}$ is first decreasing then increasing function of t^{round} over these points. This inspired us to use ternary search to quickly find a near-optimal round time, as described in Algorithm 2.

Algorithm 1 Dynamic Programming to Select Quantization Levels

Input: Number of parameters d ; Bandwidth B ; Latency per round t^{round} ; Clock frequency F_m ; Number of samples N_m ; Channel gains h_m ; Energy constraint E_m ; Transmit power $P_m, \forall m$

Output: Quantization Selections $q_m, m = 1, 2, \dots, M$;

- 1: Initialize array $Selection_M$ to store quantization selections, two-dimensional arrays $Path_{BM}$ and dp_{BM} to record state jump and dynamic program.
- 2: Calculate the value v_{mq} and weight w_{mq} for each item as formula (17a);
- 3: **for** $b = 0$ to B **do**
- 4: **for** $m = 1$ to M **do**
- 5: $\max \leftarrow 0$
- 6: **for** $q_m = 1$ to 32 **do**
- 7: **if** $b - w_{mq_m} \geq 0$ and $dp_{(b-w_{mq_m})(m-1)} + v_{mq_m} > \max$ **then**
- 8: $\max \leftarrow dp_{(b-w_{mq_m})(m-1)} + v_{mq_m}$ \triangleright Find item maximizes total value
- 9: $path_{bm} \leftarrow q_m$ \triangleright Record the quantization selection in node m
- 10: $node \leftarrow M, cap \leftarrow B$ \triangleright The current node index is M and the bandwidth is B
- 11: **while** $node > 0$ **do**
- 12: $Selection_{node} \leftarrow Path_{cap, node}$ \triangleright Fetch quantization selection
- 13: $cap \leftarrow cap - w_{node, Selection_{node}}$
- 14: $node \leftarrow node - 1,$ \triangleright Jump state to previous node
- 15: **return** $Selections_M$;

5 Experiment Results

Environment Settings. Consider $M = 8$ edge devices uniformly located in a cell of radius 500m and an edge server located at the center of the cell. Assume that all devices are re-distributed per epoch to reflect mobility. The wireless bandwidth is $B = 0.1$ MHz; path loss exponent is $\alpha = 3.76$. Set transmit power spectrum density $p_m = 20$ to 23 dBm randomly, and the power spectrum density of the channel noise is $N_0 = -174$ dBm. The maximum energy consumption of each edge device is 50 J.

Datasets and Models. We consider a federated learning task that classifies Cifar-10 data set, which has 50,000 training images and 10,000 testing images. The training samples are evenly partitioned to edge devices (i.e., 6,250 local training images for each edge device). The BA-GQ algorithm performs per epoch to accommodate device selection in FL. The computing power of edge devices is randomly selected from 10 ms/sample to 100ms/sample. We adopt LeNet model which has 62,006 parameters. All experiments are implemented on PyTorch 1.9.0 with Python 3.8, with a fixed learning rate = 0.05 and batch size = 128. The number of iterations of the training process is set to 4000.

Algorithm 2 Ternary Search for Finding Optimal Per-Round Latency

Input: Number of parameters d ; Bandwidth B ; Time division Ψ ; CPU cycle frequency F_m ; Number of samples N_m ; Channel gains h_m ; Energy constraint E_m ; Transmission power $P_m, \forall m$

Output: T^*

- 1: Give a big enough $T_{up}, T_{low} = \max\{t_m^{cp}\}, T = T_{up}$
 - 2: **while** $T_{low} + \frac{1}{\Psi} < T_{up}$ **do**
 - 3: $T_{lm} = (T_{low} + 2T_{up})/3, T_{rm} = (2T_{low} + T_{up})/3$
 - 4: Call Algorithm 1 with T_{lm} and T_{rm} and calculate $Cost_{lm}$ and $Cost_{rm}$
 - 5: **if** $Cost_{lm} < Cost_{rm}$ **then**
 - 6: $T_{up} = T_{lm}$ ▷ Optimal point is to the left of the right point
 - 7: **else**
 - 8: $T_{low} = T_{rm}$ ▷ Vice versa
 - 9: $T^* = T_{low}$
 - 10: **return** T^* ;
-

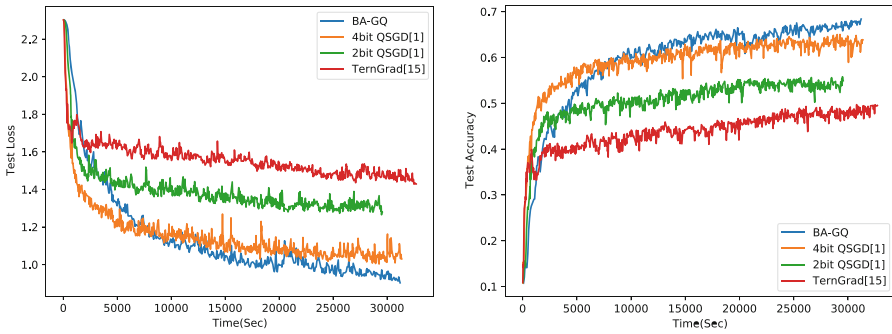


Fig. 3. Comparison with fixed quantization schemes on LeNet.

Simulation Results We compare BA-GQ algorithm with two quantization level fixed schemes TernGrad [15], QSGD [1]. We choose 2bit and 4bit quantization in QSGD as comparison objects because using more bits will violate the energy constraint. For these fixed quantization schemes, we use corresponding optimal bandwidth allocation schemes. The results are plotted in Fig. 3. From this figure, we can see that, BA-GQ algorithm outperforms QSGD and TernGrad significantly in terms of both test loss and test accuracy, when the training time is kept the same.

6 Conclusion

In this paper, we optimize the bandwidth allocation and gradient quantization for FEEL in a joint manner. By establishing a relationship between the joint optimization problem and the MCKP problem, we propose an efficient approach that can find a near-optimal solution to the joint optimization problem. Simulation results show that our proposed approach performs better than quantization level fixed schemes.

Acknowledgements. This paper is supported by the National Natural Science Foundation of China under Grant No. 61872171 and Grant No. 61832005, the Fundamental Research Funds for the Central Universities under Grant No. B210201053, the Natural Science Foundation of Jiangsu Province under Grant No. BK20190058, and the Future Network Scientific Research Fund Project under Grant No. FNSRFP-2021-ZD-07.

References

1. Alistarh, D., Grubic, D., Li, J., Tomioka, R., Vojnovic, M.: QSGD: communication-efficient SGD via gradient quantization and encoding. *Adv. Neural Inf. Process. Syst.* **30** (2017)
2. Amiri, M.M., Gunduz, D., Kulkarni, S.R., Poor, H.V.: Federated learning with quantized global model updates. *arXiv preprint [arXiv:2006.10672](https://arxiv.org/abs/2006.10672)* (2020)
3. Bernstein, J., Wang, Y.X., Azizzadenesheli, K., Anandkumar, A.: SignSGD: compressed optimisation for non-convex problems. In: *International Conference on Machine Learning*, pp. 560–569 (2018)
4. Bouzinis, P.S., Diamantoulakis, P.D., Karagiannidis, G.K.: Wireless quantized federated learning: a joint computation and communication design. *arXiv preprint [arXiv:2203.05878](https://arxiv.org/abs/2203.05878)* (2022)
5. Bubeck, S., et al.: Convex optimization: algorithms and complexity. *Found. Trends® Mach. Learn.* **8**(3–4), 231–357 (2015)
6. Chang, W.T., Tandon, R.: Mac aware quantization for distributed gradient descent. In: *IEEE Global Communications Conference*, pp. 1–6 (2020)
7. Kellerer, H., Pferschy, U., Pisinger, D.: The multiple-choice knapsack problem. In: Kellerer, H., Pferschy, U., Pisinger, D. (eds.) *Knapsack Problems*, pp. 317–347. Springer, Heidelberg (2004)
8. Konečný, J., McMahan, H.B., Yu, F.X., Richtárik, P., Suresh, A.T., Bacon, D.: Federated learning: strategies for improving communication efficiency. In: *NIPS Workshop on Private Multi-party Machine Learning* (2016)
9. Lin, X., Liu, Y., Chen, F.: Channel-adaptive quantization for wireless federated learning. In: *2021 IEEE/CIC International Conference on Communications in China (ICCC)*, pp. 457–462 (2021)
10. Liu, P., et al.: Training time minimization for federated edge learning with optimized gradient quantization and bandwidth allocation. *arXiv preprint [arXiv:2112.14387](https://arxiv.org/abs/2112.14387)* (2021)
11. McMahan, B., Moore, E., Ramage, D., Hampson, S., Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: *Artificial intelligence and statistics*, pp. 1273–1282 (2017)
12. Seide, F., Fu, H., Droppo, J., Li, G., Yu, D.: 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs. In: *Fifteenth Annual Conference of the International Speech Communication Association* (2014)
13. Tak, A., Cherkaoui, S.: Federated edge learning: design issues and challenges. *IEEE Netw.* **35**(2), 252–258 (2020)
14. Wang, Y., Xu, Y., Shi, Q., Chang, T.H.: Quantized federated learning under transmission delay and outage constraints. *IEEE J. Sel. Areas Commun.* **40**(1), 323–341 (2021)
15. Wen, W., et al.: TernGrad: ternary gradients to reduce communication in distributed deep learning. *Adv. Neural Inf. Process. Syst.* **30** (2017)



Federated Learning Meets Edge Computing: A Hierarchical Aggregation Mechanism for Mobile Devices

Jiewei Chen¹, Wenjing Li², Guoming Yang¹, Xuesong Qiu¹,
and Shaoyong Guo¹(✉)

¹ State Key Laboratory of Networking and Switching Technology,
Beijing University of Post and Telecommunications, Beijing, China
syguo@bupt.edu.cn

² State Grid Information and Communication Industry Co., Ltd.,
Beijing, China

Abstract. Federated learning (FL) has been proposed and applied in edge computing scenarios. However, the complex edge environment of wireless networks, such as limited device computing resources and unstable signals, leads to increase communication overhead and reduced performance for federated learning. Therefore, we propose a hierarchical aggregation mechanism to improve federated learning performance in a resource-constrained wireless edge environment. We propose three feature models to quantify the FL performance and design a fuzzy \mathcal{K} -means clustering mechanism. We construct an optimization problem for the process of hierarchical aggregation. And a cluster-based hierarchical federated learning algorithm (CluHFed) is designed, which consists of fuzzy clustering, asynchronous aggregation, and topology reconstruction. At last, we make an experiment with Pytorch. The results show that the proposed algorithm improves the accuracy by 2.6%–35.8% and reduces the latency of FL networks by 5.9% compared with other popular federated learning algorithms.

Keywords: Edge computing · Federated learning · Clustering · Topology reconstruction · Heterogeneity

1 Introduction

As machine learning and smart devices become increasingly popular, more and more intelligent applications are performed on the edge, such as virtual reality (VR), augmented reality (AR), image recognition, etc. [8]. Distributed model training architecture based on federated learning (FL) emerged following the great success of mobile edge computing [3].

However, in the edge wireless network environment, the performance of federated learning is restricted by heterogeneous mobile devices and unbalanced data quality, making it difficult to exert its value. The application of federated

learning in the edge environment has the following two problems that need to be solved urgently: 1) Frequent parameter transfer leads to high communication costs; 2) The wireless environment is unstable, which leads to frequent interruption or “traggler effect” in the interaction process of federated learning and affects the model quality. The problems have attracted the attention of many researchers to improve federated learning performance at the edge [5].

Several typical optimization methods have been proposed, including data edge offloading, node selection [1], model compression, etc. Ji et al. [4] proposed an Edge-Assisted Federated Learning (EAFL) mechanism, which enables stragglers to offload part of the computation to edge servers. In order to achieve a long-term performance guarantee, Xu et al. [12] formulate a stochastic optimization problem for joint client selection and bandwidth allocation under long-term client energy constraints for FL. Yin et al. [13] gave a function encryption algorithm to support multi-party data privacy-preserving and sharing. The above methods alleviated the impact of device heterogeneity and data heterogeneity on FL to a certain extent, but their help is limited.

In addition, in response to the challenge of deploying efficient federated learning tasks in resource-limited wireless edge networks, Chen et al. [1] proposed a federated learning framework for joint communication and learning, in which joint learning, wireless resource allocation, and user selection problem are formulated as an optimization problem. To deal with edge dynamics, Liu et al. [6] proposed an adaptive asynchronous federated learning (AAFL) mechanism, in which the parameter server will aggregate local updated models only from a certain fraction of all edge nodes in each epoch. These works endow FL algorithms with wireless communication properties and quantify the impact of wireless factors on FL performance [10]. However, as far as our survey is concerned, there is still a lack of research on the topology control of federated learning networks. It is expected to come up with FL solutions that take into account training efficiency, stability, and continuity.

To address this issue, we propose a cluster-based hierarchical aggregation mechanism for FL. We construct an edge-assisted clustering framework for federated learning (EFL), including energy consumption modeling, resource alignment modeling, and data heterogeneity modeling. Then we propose a fuzzy clustering mechanism to discover the optimal cluster. To ensure the stability of the mobile clients, we build an FL network topology reconstruction model for a timely response. In addition, we use an asynchronous hierarchical aggregation mechanism based on delay evaluation to adapt to the high dynamics of mobile devices. From this, we construct a resource-constrained clustering optimization problem for the EFL network that jointly minimizes the loss of accuracy and the sum of squared errors within the clusters. And we design a cluster-based hierarchical FL algorithm including three modules to solve this problem. At last, we make an experiment with Pytorch on two datasets. The experimental results show that compared with the typical FL algorithms, our proposed scheme reduces communication costs and improves the accuracy in an unstable environment.

2 System Structure

2.1 Edge-Assisted Hierarchical Federated Learning Architecture

Figure 1 illustrates our proposed cluster-based hierarchical federated learning architecture in wireless networks. The three-layer architecture of federated learning networks is consist of three components: client, cluster center (CC), and parameter server (PS), which are deployed in mobile devices, edge server, and cloud respectively.

- Device layer: the clients form a service topology for faster detection of stragglers (i.e., the federated learning workers who need to wait for a long time). The clients perform local iterations to train a local modal and send the model parameters to the CC for edge aggregation.
- Edge layer: all the CCs aggregate the local model parameters collected within a cluster and update edge models. Besides, the CCs are responsible for forwarding the collected resource information and edge model to the PS.
- Cloud layer: the PS will aggregate the edge model parameters asynchronously for updating the global model, and evaluate the operational quality and resource consumption of the federated learning network.

2.2 Cluster-Based Hierarchical Federated Learning Workflow

The architecture also contains three functional modules: fuzzy clustering, asynchronous updating, and topology reconstruction.

1) Fuzzy Clustering. Before federated training, according to the resource state and data distribution of each client, the energy consumption, training time, and data distribution characteristics are extracted, and the nodes are divided into \mathcal{K} clusters by fuzzy clustering method.

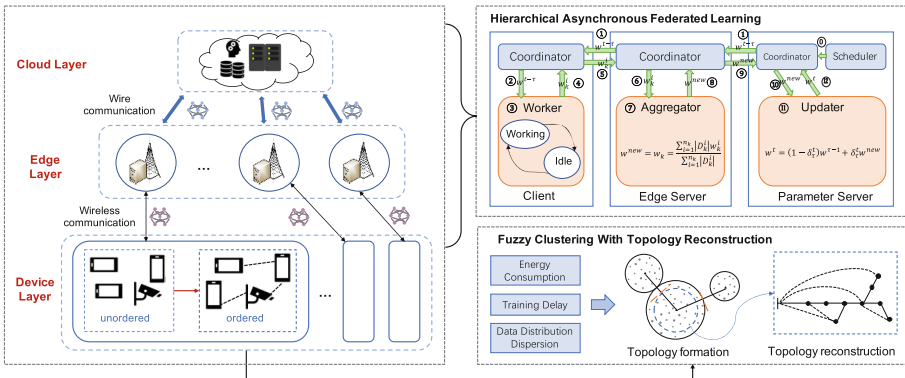


Fig. 1. Cluster-based hierarchical federated learning architecture with topology reconstruction.

2) Hierarchical Asynchronous Federated Learning. Client i in cluster k use gradient descent to train the local model. And it sends updated local models w_i^k to edge nodes, and edge nodes are responsible for collecting all local models of cluster k where they are located, and perform the first aggregation, which is called “edge aggregation”. Then the model is uploaded to the aggregator in the cloud. At the same time, the edge node will broadcast w_k to the clients in cluster k to perform the next iteration of the local model training. The edge node does not access the local raw data D_k^i of each client i during this process. After each edge node, $EN_k \in \mathcal{K}$ completes the edge aggregation and uploads w_k to the cloud, the cloud server will accept all the models from the edge and perform global aggregation.

3) Topology Reconstruction. In order to adapt to the unstable wireless environment, the graph model of mobile devices in the federated learning network is established, and the heterogeneous device database in a fuzzy hierarchical space is constructed. The reconstructed topology will be sent to the cloud center for the next round of federated learning updates.

3 Cluster-Based Federated Learning Networks

3.1 Feature Model

Assuming that there are N clients willing to participate in federated learning training, and the PS cluster the clients according to the collected features V_i . The features include the remaining energy of client, the distance from CC (sink node), and the gap between single data distribution and the overall distribution. We use the fuzzy clustering method for initial stratification to determine the number of clusters \mathcal{K} .

Energy Consumption Model. The energy consumption of the client is mostly used for data transfer and computation of datasets. Let the energy consumed by client i to transmit g -bit data to j is

$$E_{tran}(i, j) = g \times \alpha_{del} + g \times \alpha_{op} \times d_{ij}^\gamma \quad (1)$$

where α_{del} represents the unit energy consumed by the data processing circuit, α_{op} represents the unit energy consumed by the running circuit, d_{ij} represents the distance between clients i and j , γ is path attenuation index. For one local iteration, the energy consumption of client i is expressed as

$$E_{cop} = \varsigma c_i \beta_i f_i^2 \quad (2)$$

The CPU cycle frequency of client i is expressed as f_i , the number of CPU cycles required to train the local model is c_i , β_i is the sample data size, and ς denotes the effective capacitance parameter of the chipset.

Resource Alignment Model. The PS connects to the resource pool of each AP for clients at the beginning of federated learning, and the resource information is used to estimate the P2P transmission throughput under a specific topology. When a new round of calculation starts, the throughput of the client's P2P link under different topologies is estimated, and finally the throughput is converted into the required communication time during aggregation.

Denote the transfer rate of parameters as $r_k = B \ln \left(1 + \frac{\rho_k h_k}{N_0} \right)$, where B is the transmission bandwidth, ρ_k is the transmission power of the CC k , h_k is the channel gain of the P2P link between the CC k , and N_0 is the possible noise. Let the number of model parameters obtained by edge aggregation of CCs be σ , which is a definite value. Then the time for CC k to transmit model parameters is

$$T_k^{com} = \frac{\sigma}{\left[B \ln \left(1 + \frac{\rho_k h_k}{N_0} \right) \right]} \quad (3)$$

Denote the communication time between clients as H_i^{com} , and use $|\omega|$ to represent the size of the local training model ω , then we have

$$T_i^{com} = 2 \cdot \frac{|\omega|}{TP_i}, \quad (4)$$

where TP_i represents the P2P communication throughput between clients, which is determined by the network topology and the efficiency of the federated learning algorithm.

Data Heterogeneity Model. For the non-IID data in the federated learning network, we use the EMD distance as one of the clustering features to ensure the training effect of federated learning [14]. Assuming that the training data of the i -th client is sampled from IID in the distribution \mathcal{D}_i , then the overall distribution can be considered to be a mixture of all local data distributions, that is $\mathcal{D} = \sum_{i=1}^n p_i \mathcal{D}_i$. We define the SGD weight divergence of federated learning as

$$\text{Weight divergence} \triangleq \frac{\|w^{avg} - w^{SGD}\|}{w^{SGD}} \quad (5)$$

Let $p(y = z)$ be the overall probability distribution, and $p_i(y = z)$ be the local probability distribution of the i -th client, then the EMD distance of client i is

$$EMD(p, p_i) = \sum_{i=1}^n \|p_i(y = z) - p(y = z)\| \quad (6)$$

3.2 Federated Learning Network Topology Model

To achieve fast response and efficient processing of complex federated learning tasks, we build a federated learning network topology model. When it occurs time-varying working conditions, node failure, or insufficient number of clients,

the idle clients can join the task at any time and the network topology would be reconstructed.

The federated learning network is represented by an undirected graph $G(S, D, N)$, where $S = \{s_0, s_1, \dots, s_n\}$, n indicates the number of clients in the network, and $n = |S| - 1$. D is the distance set between any two points (clients), and also represents the edge set formed by the client in G . Then we have

$$D = \{d(s_i, s_j) | \forall i, j \leq n\} \quad (7)$$

where $d(s_i, s_j)$ represents the distance between s_i and s_j . Let N_i denotes the one-hop neighbor node set of s_i , defined as follows:

$$N_i = \{s_{ij} | d(s_i, s_j) \leq r, \forall i \neq j, s_i \in S\} \quad (8)$$

In this formula, r represents the communication radius of one hop, so N also represents the adjacency matrix of G . For two clients $s_i, s_j \in S$, if there is $d(s_i, s_j) \in D$, then $s_{ij} = 1$, called s_i and s_j can communicate; if $d(s_i, s_j) \notin D$, then $s_{ij} = 0$. Define the Laplace matrix $L(G) = [l_{ij}]_{n \times n}$ to satisfy $L(G) = D - N$, where

$$l_{ij} = \begin{cases} \sum_{i \neq j} s_{ij}, & i = j \\ -s_{ij}, & i \neq j \end{cases}$$

Let the state of the federated learning network satisfy the following dynamic equation:

$$\tilde{x}_i(t) = Ax_i(t) + Bu_i(t) \quad (9)$$

where $\tilde{x}_i(t) \in R^n$ is the state of the client i , $u_i(t) \in R^c$ is the control input of the network, A and B are coefficient matrices. When the communication network between clients is connected, it can be considered that the client joins the FL network. When the communication network is not connected, we need to reconstruct the communication network topology.

Definition 1. Let $G = (S, D, N)$ be a directed graph of order n , define a matrix $P = (p_{ij})_{n \times n}$, that is

$$p_{ij} = \begin{cases} 1, & \text{there is at least one directed chain from } s_i \text{ to } s_j \\ 0, & \text{there is no directed chain from } s_i \text{ to } s_j \end{cases} \quad (10)$$

Call P the reachability matrix of directed graph G .

When the order of G is large ($n > 50$), the reachability matrix can also be directly calculated through the adjacency matrix N , i.e.,

$$p_{ij} = \begin{cases} 1, & l_{ij}^{(n)} > 0 \\ 0, & l_{ij}^{(n)} = 0 \end{cases}$$

4 Cluster-Based Hierarchical Federated Learning Mechanism with Topology Reconstruction

4.1 Problem Formulation

In this section we propose a Cluster-based and Resource-Constrained Federated Learning Problem (CRC-FLP) to find the optimal cluster structure \mathcal{K} . We consider the resources related to computation and communication. For a specific federated learning task, we want the final trained model to perform best, and the completion time of training can be shortened to achieve efficient and feasible federated learning, which directly reflects the loss value of the global model and the error between the cluster center and the clients. We write the following multi-objective programming problem:

$$\min_{\mathcal{K} \in \{1,2,3,\dots,N\}} P_1 F(w^f) + P_2 \sum_{k=1}^{\mathcal{K}} \sum_{i \in S} |x_i - \mu_k|^2 \quad (11)$$

$$\text{s.t. } C1 : \sum_{t=1}^T n_k p_k^t (E_{tran}^k + E_{cop}) \leq E_{max}, \forall k, \quad (12)$$

$$C2 : \sum_{k=1}^{\mathcal{K}} \sum_{i \in S} (T_i^{com} + T_k^{com}) \leq T_{max}, \quad (13)$$

$$C3 : \sum_{k=1}^{\mathcal{K}} p_k^t = 1, \forall t, \quad (14)$$

$$C4 : n_k = \frac{N}{\mathcal{K}}, \forall k. \quad (15)$$

P_1 and P_2 in the objective function respectively represent the priority factors of the two objectives. $\sum_{k=1}^{\mathcal{K}} \sum_{i \in \mathcal{K}} |x_i - \mu_k|^2$ represents the sum of squared errors (SSE) after clustering, where x_i denotes each scattered point and μ_k is the centroid. We use E_{cop}^i to represent the resources consumed by the clients to perform a local iteration, and E_{cop}^k to represent the resources consumed by the edge nodes to perform an edge aggregation. C1 indicates that the total energy consumption of T iterations does not exceed the energy budget, where n_k is the number of client in cluster k , and p_k^t indicates whether cluster k has joined the global aggregation in t -th iteration. C2 represents the delay tolerance of hierarchical FL. C3 means that all clusters perform global updates asynchronously. C4 illustrates the relationship between the cluster structure and the number of client.

4.2 Cluster-Based Hierarchical Federated Learning Algorithm (CluHFed)

Obviously, it is impossible to solve the CRC-FLP directly. In the literature [10], it is proved that the convergence property of hierarchical federated learning is

Algorithm 1 CluHFed Algorithm

Input: resource budget E_{max}, T_{max} ; the number of clients N
Output: federated learning model parameter w^f

- 1: **Fuzzy \mathcal{K} -Means Clustering at PS:**
- 2: **for** the number of iterations b **do**
- 3: calculate the partition matrix $U = [u_{ij}]_{n \times n}$ and select the cluster center $\mu_k(b)$;
- 4: $b = b+1$;
- 5: **if** $\|\mu_k(b) - \mu_k(b-1)\| < \epsilon$ **then** # Determine whether to converge.
- 6: **return** cluster center μ_k .
- 7: **Federated Learning with Hierarchical Aggregation:**
- 8: initialize $w_0 = 0, t = 0$;
- 9: **repeat**
- 10: # Local training at clients:
- 11: compute w_i^k by $w_i^k(t) = w_i^k(t-1) - \eta \nabla F_i(w_i^k(t-1))$;
- 12: # Edge aggregation at cluster centers (CCs):
- 13: compute w_k by $w_k(t) = \frac{\sum_{i=1}^N D_i w_i^k(t)}{D}$;
- 14: record resource consumption E_{cop} and T_i^{com} ;
- 15: send $w_k, \sum_{i=1}^{n_k} T_i^{com}$ and E_{cop} to the PS;
- 16: # Global aggregation at PS:
- 17: compute w_t according to formula (17); estimate E_{tran}^k and T_k^{com} ;
- 18: update current resource consumption $E(t)$ and $T(t)$;
- 19: Send w_t back to the CCs;
- 20: **until** $E > E_{max}$ or $T > T_{max}$.
- 21: **return** the final model parameter w^t .
- 22: **Topology Reconstruction at CCs:**
- 23: initialize adjacency matrix N_s ;
- 24: **repeat**
- 25: **for** $i = 1$ **do**
- 26: **if** $N(j, i) == 1 (j = 1, 2, \dots, n)$ **then**
- 27: $N(j, k) = N(j, k) \vee N(i, k), (k = 1, 2, \dots, n)$;
- 28: **until** all elements in N_s are 1 #i.e., the current network topology is connected
- 29: send s_i to the PS for re-clustering.

related to the number of local iterations of the client. Therefore, we propose a *Cluster-based Hierarchical Federated Learning (CluHFed)* Algorithm consisting of three modules to solve the problem model.

First, we use the fuzzy clustering method to determine the initial value of \mathcal{K} (lines 1–6). According to the feature models for the clients, we can get a feature space \mathcal{V} . Then we transform the clustering problem into the following nonlinear programming problem:

$$\min J = \sum_{k=1}^{\mathcal{K}} \sum_{i=1}^n u_{ki} \|x_i - \mu_k\|^2 \quad (16)$$

where J is the minimization target, i.e., the squared error in the cluster, $\mathcal{K} \in \{1, 2, 3, \dots, N\}$, μ_k denotes the cluster center. And u_{ki} forms a \mathcal{K} -cluster

partition matrix, which represents the probability that the scatter point (client) i is divided into cluster k . There are $n \times k$ groups of data in the matrix, each group of data contains v columns, representing v -dimensional vectors.

Then, our proposed asynchronous update mechanism is used for federated learning training (lines 7–21). For cluster k , we use τ to denote the number of local iterations of its internal clients since the last global update, then $(t - \tau)$ denotes the interval rounds of trained model in t -th iteration [9]. The model weights received from any cluster will be determined by τ , i.e., the global update rule for the cloud server is:

$$w^t = (1 - \delta_\tau^t) w^{t-1} + \delta_\tau^t w_k \tag{17}$$

where w_k represents the edge model parameters received in t -th iteration, and δ_τ^t represents the weight of the edge model w_k in t -th global iteration. We use the function in [2] to determine the value of δ_τ^t .

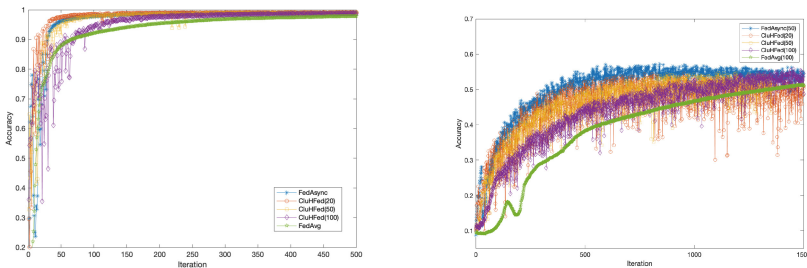
Finally, we design an improved Warshell algorithm to generate and reconstruct the communication topology (lines 22–29), which means the PS will re-evaluate the clients’ status after each global aggregation. The specific algorithm flow is described in Algorithm 1.

5 Simulation Experiment Evaluation

5.1 Experimental Setting

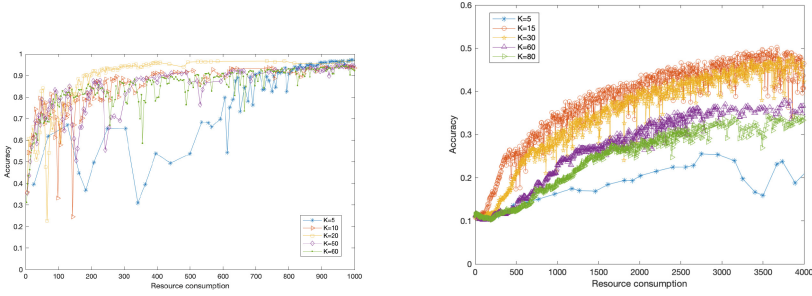
We build the hierarchical asynchronous federated learning training framework based on the Pytorch and verified our proposed algorithm on popular image datasets MNIST and CIFAR-10, where the CNN model is mainly used. We also compare the accuracy, computation completion time and resource consumption with several popular FL algorithms (*FedAvg* [7], *FedAsync* [11]).

We set the resource consumption between CC and PS to unit 1, then the communication consumption between clients is 0.1, because the communication delay between CC and PS is ten times that of intra-cluster communication.



(a) Accuracy vs. No. of iteration over MNIST. (b) Accuracy vs. No. of iteration over CIFAR10.

Fig. 2. Convergence evaluation of the algorithms.



(a) Accuracy vs. resource consumption over MNIST. (b) Accuracy vs. resource consumption over CIFAR10.

Fig. 3. Stability evaluation of CluHFed.

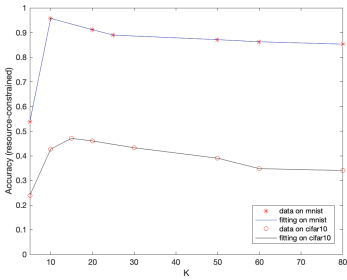


Fig. 4. The fitting curve of accuracy under different \mathcal{K} .

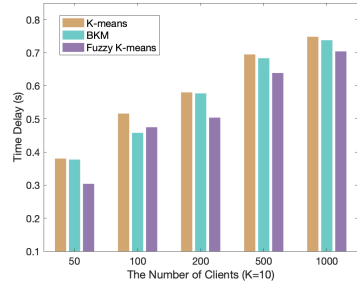


Fig. 5. The time delay of three algorithms.

Meanwhile, we adopt an asynchronous training mechanism and design a priority queue, only one cluster uploads its aggregation parameters at a time slot. We set the learning rate $\eta = 0.01$, local iteration $h = 10$ times in each global aggregation. In order to evaluate the effectiveness of *CluHFed*, we set different number of clusters $\mathcal{K} = \{5, 10, 15, 20, 25, 50, 60, 80\}$ to observe the impact on FL performance. We also test the effect of clustering and topology reconstruction on federated learning.

5.2 Experimental Results

We first verify the convergence of the proposed algorithm. Figure 2 shows the comparison between the *CluHFed* algorithm and the benchmark algorithm when the number of clients is 20, 50, and 100, respectively. It can be seen in Fig. 2(a), in the early stage, the convergence speed of *CluHFed* is about 20 rounds faster than that of FedAvg, 10 rounds faster than that of FedAsync. And *CluHFed* is about 300 rounds faster than FedAvg in reaching the same convergence accuracy. In Fig. 2(b), the convergence speed of *CluHFed* is about 500 rounds faster than that of FedAvg, and is comparable to that of FedAsync.

As shown in Fig. 3, it can be seen that whether it is MNIST or CIFAR10, there is always an optimal \mathcal{K}^* value that makes the algorithm achieve the best convergence performance. In Fig. 3(a), CluHFed has the best convergence performance when $\mathcal{K} = 20$, the optimal number of clusters can improve accuracy by 5.4%–31.2% when resources are limited. In Fig. 3(b), CluHFed has the highest accuracy when $\mathcal{K} = 15$, the optimal number of clusters can improve accuracy by 2.6%–35.8% when resources are limited. Further, when the size of nodes increases, CluHFed exhibits stronger convergence stability and lower volatility. This shows that the topology reconstruction based on heterogeneous state space has a good effect on the training stability of federated learning.

We fit the convergence accuracy under a different number of clusters into a curve, as shown in Fig. 4, the fitting curve is an approximate convex function, which further verifies the validity of our algorithm.

For clustering experiments, we compare the time delay with K-means and BKM. We determined the number of clusters to be 10 for the different number of clients. The clustering experiment takes the MNIST dataset as an example, in the case that the same class is divided into the same cluster, the FL time delay is compared and analyzed. It can be seen from the Fig. 5 that the proposed fuzzy clustering method can always maintain a high aggregation efficiency. Taking the number of 200 as an example, compared with the K-means algorithm and the BKM algorithm, the fuzzy \mathcal{K} -means algorithm reduces the latency of FL by 5.9% and 4.6%, respectively.

6 Conclusion

In this paper, we propose a hierarchical aggregation mechanism for federated learning, including fuzzing clustering, asynchronous updating, and topology reconstruction. We build a cluster-based hierarchical federated learning architecture on which we implement our proposed mechanism and the algorithm. We demonstrate the benefit of the proposed mechanism and the effectiveness of the CluHFed algorithm by the experimental results. It is proved that the CluHFed algorithm with fuzzy \mathcal{K} -means outperforms the typical algorithms. In future work, we plan to study multi-hop routing in federated learning network topology and build a more robust hierarchical federated learning architecture.

Acknowledgments. This work was supported by the National Key R&D Program of China (2020YFB2104503) and the National Natural Science Foundation of China (No. 62071070).

References

1. Chen, M., Yang, Z., Saad, W., Yin, C., Poor, H.V., Cui, S.: A joint learning and communications framework for federated learning over wireless networks. *IEEE Trans. Wirel. Commun.* **20**(1), 269–283 (2021)

2. Chen, Y., Ning, Y., Slawski, M., Rangwala, H.: Asynchronous online federated learning for edge devices with non-IID data. In: 2020 IEEE International Conference on Big Data (Big Data), pp. 15–24 (2020). <https://doi.org/10.1109/BigData50022.2020.9378161>
3. Deng, S., Zhao, H., Fang, W., Yin, J., Dustdar, S., Zomaya, A.Y.: Edge intelligence: the confluence of edge computing and artificial intelligence. *IEEE Internet Things J.* **7**(8), 7457–7469 (2020)
4. Ji, Z., Chen, L., Zhao, N., Chen, Y., Wei, G., Yu, F.R.: Computation offloading for edge-assisted federated learning. *IEEE Trans. Veh. Technol.* **70**(9), 9330–9344 (2021)
5. Lim, W.Y.B., et al.: Decentralized edge intelligence: a dynamic resource allocation framework for hierarchical federated learning. *IEEE Trans. Parallel Distrib. Syst.* **33**(3), 536–550 (2022)
6. Liu, J., et al.: Adaptive asynchronous federated learning in resource-constrained edge computing. *IEEE Trans. Mob. Comput.* **1** (2021)
7. McMahan, B., Moore, E., Ramage, D., Hampson, S., Arcas, B.A.Y.: Communication-efficient learning of deep networks from decentralized data. In: Singh, A., Zhu, J. (eds.) Proceedings of the 20th International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 54, pp. 1273–1282. PMLR (2017)
8. Qiu, C., Wang, X., Yao, H., Xiong, Z., Yu, F.R., Leung, V.C.M.: Bring intelligence among edges: a blockchain-assisted edge intelligence approach. In: GLOBECOM 2020–2020 IEEE Global Communications Conference, pp. 1–6 (2020)
9. Wang, S., et al.: Adaptive federated learning in resource constrained edge computing systems. *IEEE J. Sel. Areas Commun.* **37**(6), 1205–1221 (2019). <https://doi.org/10.1109/JSAC.2019.2904348>
10. Wang, Z., Xu, H., Liu, J., Huang, H., Qiao, C., Zhao, Y.: Resource-efficient federated learning with hierarchical aggregation in edge computing. In: IEEE INFOCOM 2021 - IEEE Conference on Computer Communications, pp. 1–10 (2021)
11. Xie, C., Koyejo, S., Gupta, I.: Asynchronous federated optimization (2019). <https://doi.org/10.48550/ARXIV.1903.03934>, <https://arxiv.org/abs/1903.03934>
12. Xu, J., Wang, H.: Client selection and bandwidth allocation in wireless federated learning networks: a long-term perspective. *IEEE Trans. Wirel. Commun.* **20**(2), 1188–1200 (2021)
13. Yin, L., Feng, J., Xun, H., Sun, Z., Cheng, X.: A privacy-preserving federated learning for multiparty data sharing in social IoTs. *IEEE Trans. Netw. Sci. Eng.* **8**(3), 2706–2718 (2021)
14. Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., Chandra, V.: Federated learning with non-IID data (2018). <https://doi.org/10.48550/ARXIV.1806.00582>



QoS-oriented Hybrid Service Scheduling in Edge-Cloud Collaborated Clusters

Yanli Ju¹, Xiaofei Wang¹(✉), Xin Wang¹, Xinying Wang², Sheng Chen², and Guoliang Wu³

¹ College of Intelligence and Computing, Tianjin University, Tianjin, China
{yanliju,xiaofeiwang,wangx}@tju.edu.cn

² China Electric Power Research Institute, Beijing, China
{wangxinying,chensheng}@epri.sgcc.com.cn

³ Hei Longjiang Electricity Power Company of State Grid, Harbin, China

Abstract. Service scenarios under edge-cloud collaboration are becoming more diverse in terms of service performance requirements. For example, smart grids require both intelligent control and long-term optimization, which poses considerable challenges for service providers to meet quality of service (QoS). However, current pioneering work has not yet explored both system utility and QoS guarantees. Therefore, this paper investigates the optimization problem of edge-cloud collaborative scheduling for QoS guarantees. First, we model the edge-cloud collaborative scheduling scenario and derive two sub-problems such as service deployment and request dispatch. Second, we design a near-optimal scheduling algorithm based on a submodular function optimization approach with the objective of maximizing the number of requests that are processed within the edge-cloud cluster under QoS constraints. Finally, our experiments verify the beneficial effects of the proposed algorithm in terms of throughput rate, scheduling time cost, and resource utilization.

Keywords: Edge-cloud collaboration · Edge computing · Service deployment · Request dispatch

1 Introduction

Edge computing moves computing and storage capabilities down to the edge of the network close to the data source, which can effectively reduce congestion in the backbone network while significantly reducing transmission latency and improving Quality of Service (QoS) [14]. The massive resources of cloud computing are more suitable for long-period, non-real-time big data processing and analysis. It can be seen that edge computing and cloud computing do not belong to a mutual substitution relationship, but should complement each other's capabilities in the form of edge-cloud collaboration. Therefore, the joint scheduling of edge-cloud resources and tasks based on service deployment and request dispatch can further amplify the beneficial complementary effects of edge and cloud, and thus improve the throughput of edge-cloud cluster systems [9].

However, scheduling for edge-cloud clusters, such as smart grid clusters, is not a breeze. In the 5G era, the service scenarios will have more and more diverse requirements in service performance, which imposes a greater challenge to meet the QoS [8]. Most of the current research is oriented towards QoS for resource deployment, and solutions that focus on **both system utility and QoS guarantees** are still missing. According to different QoS requirements, service can be classified into two categories [1]: 1) Latency-critical (LC) services that must meet strict QoS guarantees, such as virtual reality. 2) Best-effort Batch (BE) services, such as data analysis. If these two types of services and requests are to be scheduled uniformly, it will not only cause resource competition within the LC services or BE services, but also cause shared resource competition, which leads to service execution interference and performance loss. Since such interference is particularly disruptive for LC services, edge clusters need to be able to dynamically schedule service deployments and requests [3, 11] under resource constraints to meet the QoS requirements and to maximize throughput.

Therefore, for the two main scheduling requirements such as service deployment and request dispatch in edge-cloud clusters, this paper investigates the optimization of edge-cloud collaborative scheduling and take smart grid clusters as an example, with the aim of improving system utility while guaranteeing QoS. First, this paper **designs a service deployment algorithm** with the objective of maximizing the number of requests processed under QoS constraints. Second, this paper **employs dual time-scale scheduling** to coordinate service deployment and request dispatch for reducing scheduling overhead. Finally, this paper **designs a two-tier mechanism** to schedule the LC and BE services for improving the utilization of resources.

The main contributions of this paper are summarized as follows: 1) We propose an QoS-oriented edge-cloud collaborative scheduling architecture, which employs dual time-scale scheduling to coordinate service deployment and request dispatch, and designs a two-tier mechanism to schedule LC and BE services to improve the utilization of resources. 2) We perform a complexity analysis of the above joint scheduling optimization and prove that it is NP-hard; meanwhile, we design an algorithm, Service Deployment based on Submodular Function Maximization (SD-SFM), for solving this problem. 3) We conduct real dataset-driven experiments on the proposed architecture and algorithm, selecting scheduling time consumption, resource utilization, and throughput rate as metrics, and verify the effectiveness of the scheduling solution using experiments.

The remainder of this paper is organized as follows. Related works are discussed in Sect. 2 and Sect. 3 elaborates the system model and the schedule problem. Then, the proposed solutions are derived in Sect. 4 along with experiment results given in Sect. 5. Finally, Sect. 6 concludes the paper.

2 Related Work

Some research works [9] have established the theoretical basis for the joint optimization of service deployment and request dispatch. Liang et al. [7] modeled

the service deployment problem of edge-cloud clusters and designed a greedy-based service deployment algorithm. However, Liang et al.'s work did not directly consider the dynamic and competitive nature of resources in actual deployment. Ning et al. [11] fully considered the dynamic nature of service deployment scenarios, decomposed the long-term optimization problem into a series of immediate optimization problems using the Lyapunov optimization method, and used a distributed Markov approximation algorithm to determine the service deployment (placement) configuration.

Further, Hudson et al. [6] studied joint service placement and model scheduling of edge intelligence services with the goal to maximize QoS for end users where edge intelligence services have multiple implementations to serve user requests, each with varying costs and QoS benefits. Chen et al. [1] presented a QoS-aware resource manager that enables an arbitrary number of interactive, latency-critical services to share a physical node without OoS violations. The proposed resource manager leverages a set of hardware and software resource partitioning mechanisms to adjust allocations dynamically at runtime, in a way that meets the QoS requirements of each co-scheduled workload, and maximizes throughput for the machine. Nevertheless, none of the above pioneering works on scheduling for edge-cloud clusters consider the QoS guarantees and the hybrid deployment of LC and BE services.

Table 1. Table of notations

Symbol	Definition	Symbol	Definition
\mathcal{L}	The set of LC services	\mathcal{N}	The set of edge nodes
x_{ln}	Deployment variables for LC services	y_{lnm}	Dispatch variables for LC service requests
r_l	Storage requirement for LC service l	R_n	Storage capacity of edge node n
k_l	Computation requirement per request of LC service l	K_n	Processing capacity of edge node n
R'_n	Remaining storage capacity of edge node n after deploying LC services	K'_n	Remaining processing capacity on edge node n after processing LC service requests
λ_{ln}	Average arrival rate of LC service requests	T_l	Maximum tolerated response time of service l
O_l	Processing time per request of service l	t_{nm}	Transmission delay from edge node n to m

3 Scenario Description and Problem Modeling

In this section, we present the scheduling problem of edge-cloud clusters under QoS guarantees and take smart grid clusters as an example. We first describe a system framework to elaborate service deployment and request dispatch. Then, we formulate a scheduling problem of smart grid clusters towards a single exemplary QoS requirement, while adding constraints to such model to guarantee QoS requirements.

3.1 System Model

As shown in Fig. 1, there is an smart grid cluster system on a geographic region where end devices are covered by a set of edge clusters. The smart grid cluster

system considered in this paper includes an edge cluster and a cloud cluster, where the edge cluster consists of a set of edge nodes $\mathcal{N} = \{1, \dots, N\}$ for smart grids. Each edge node for smart grids has limited storage and computing resources, and the resource capacities among the edge nodes for smart grids are heterogeneous with each other. All the services undertaken by the smart grid cluster include LC services $\mathcal{L} = \{1, \dots, L\}$ and BE services $\mathcal{L}' = \{1, \dots, L'\}$. Meanwhile, service heterogeneity is not only reflected in terms of resource requirements, but also in terms of QoS requirements. The symbols involved in this paper and their meanings are given in Table 1.

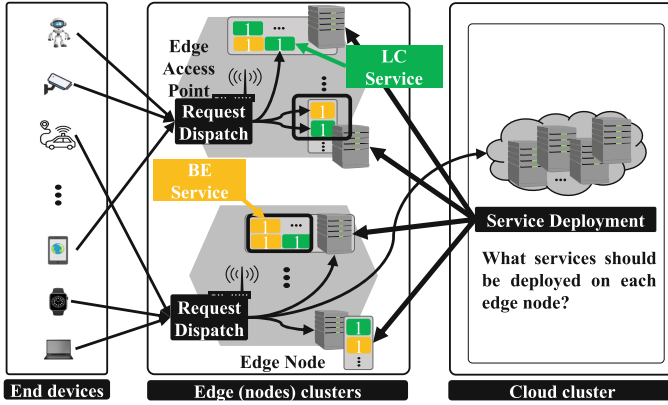


Fig. 1. Edge-cloud smart grid cluster scheduling for hybrid service deployments.

3.2 Problem Formulation

The heterogeneity of service requirements requires differentiated QoS guarantees. To meet the different QoS guarantees, Liu et al. designed a resource channel pre-configuration (slicing) solution “*EdgeSlice*” [8]. *EdgeSlice* initializes the resource channel configuration according to the different QoS requirements of services, i.e., allocates resource units composed of compute, storage, and network resources to services. The QoS-oriented scheduling optimization problem with multiple QoS constraints can be decoupled into a scheduling optimization problem with multiple single QoS guarantees, provided that such solutions have been pre-configured with effective resource channels. Therefore, this paper focuses on the resource channel corresponding to a certain QoS requirement (characterized by the “maximum tolerated response time” property) without losing the generality for other resource channels.

Considering that the scheduling optimization problem discussed in this work includes both service deployment and request dispatch, we set two decision variables: the service deployment variable x_{ln} and the request assignment variable y_{lnm} . If a replica of service l is deployed on edge node n , x_{ln} takes the value of 1 otherwise 0. Since the meaning of y_{lnm} is the probability that a request of service l submitted to an edge access point n is dispatched to an edge node

m , the value of y_{lnm} ranges from $[0,1]$. In this case, the scheduling optimization problem of LC services can be modeled as problem (1), whose objective function is to maximize the throughput of LC services, i.e., the number of LC requests that are successfully dispatched under guaranteed QoS requirements.

$$\max : \sum_{l \in \mathcal{L}} \sum_{n \in \mathcal{N}} \lambda_{ln} \sum_{m \in \mathcal{N}} y_{lnm}, \quad (1a)$$

$$\text{s.t.} : \sum_{m \in \mathcal{N}} y_{lnm} \leq 1, \quad \forall l \in \mathcal{L}, n \in \mathcal{N}, \quad (1b)$$

$$\sum_{l \in \mathcal{L}} x_{ln} \cdot r_l \leq R_n, \quad \forall n \in \mathcal{N}, \quad (1c)$$

$$\sum_{l \in \mathcal{L}} k_l \sum_{n \in \mathcal{N}} \lambda_{ln} \cdot y_{lnm} \leq K_m, \quad \forall m \in \mathcal{N}, \quad (1d)$$

$$y_{lnm} \leq I_{\{T_l - O_l - t_{nm}\} > 0} \quad \forall l \in \mathcal{L}, n \in \mathcal{N}, m \in \mathcal{N}, \quad (1e)$$

$$y_{lnm} \leq x_{lm}, \quad \forall l \in \mathcal{L}, n \in \mathcal{N}, m \in \mathcal{N}, \quad (1f)$$

$$x_{ln} \in \{0, 1\}, y_{lnm} \geq 0, \quad \forall l \in \mathcal{L}, n \in \mathcal{N}, m \in \mathcal{N}. \quad (1g)$$

We jointly optimize service deployment and request dispatch by a dual time-scale scheduling mechanism. The dual time-scale refers to the larger time scale “frame” and the smaller time scale “slot”, where a time frame contains multiple time slots. At each time slot, the system performs request dispatch, so there are requests arriving at the edge access point waiting for dispatch, i.e., the request arrival rate $\lambda = (\lambda_{ln})$ is obtained at each slot, where λ_{ln} denotes the request arrival rate of service l submitted to edge access point n . At the granularity of each time frame, it performs service deployment. Considering the deployed services are fixed over the whole frame, the request arrival rate parameter λ in problem (2) should consider the whole time frame range. According to the work of Farhadi et al. [3], the average request arrival rate that can be predicted is considered as a known parameter in this work. In this manner, the system can periodically adjust the service deployment strategies to ensure that the deployed services match the current pattern of request arrivals. In addition, this dual time-scale design can reduce scheduling overhead compared to performing request dispatch and service deployment at the same frequency [3].

With the above premises, the joint optimization problem on LC service deployment and request dispatch can be formulated as problem (1). Objective (1a) is to maximize the number of LC requests processed under the QoS constraint. Constraint (1b) ensures the decision variables for request dispatch are valid. Constraint (1c) guarantees that the edge node for smart grids does not deploy more services than its storage capacity. Constraint (1d) ensures that each edge node for smart grids does not process more requests than its computing capacity. Constraint (1e) indicates that a request is dispatched successfully provided that the QoS is guaranteed, i.e., the sum of the process time and the transmission delay of the request will not exceed the maximum response time that the service can tolerate, where $I_{\{T_l - O_l - t_{nm}\} > 0}$ is the indicator function. Constraint (1f) indicates that a request is dispatched successfully provided that

the requested service is deployed on that edge node for smart grids. Constraint (1g) specifies the valid value ranges of the scheduling variables.

To improve resource utilization without affecting the utility of LC services, the paper designs to run BE services on the remaining resources after LC service scheduling. The optimization objective of BE service scheduling is consistent with that of the LC, i.e., to maximize the throughput of BE services. The constraints of BE service scheduling are also formally identical with problem (1), but it is noteworthy that available resources for BE requests are not constrained by R_n, K_n , but R'_n, K'_n . Given that both LC and BE service scheduling optimization can be expressed in the form of problem (1), only the solution to problem (1) is explored below. Since it involves both integer variables $\mathbf{x} = (x_{ln})$ and real variables $\mathbf{y} = (y_{lnm})$, problem (1) is clearly a mixed-integer linear programming problem. Since mixed-integer linear programming is an NP-hard problem [10], problem (1) is an NP-hard problem.

4 Two-Tier Scheduling for Edge-Cloud Clusters

In this section, we design a scheduling algorithm based on submodular function optimization with the objective of maximizing the number of successfully processed requests, and prove the near-optimality of the algorithm theoretically.

4.1 Two-Tier Scheduling Algorithm

We further decompose problem (1) into two subproblems, service deployment and request dispatch, and design scheduling optimization algorithms for each of them. Since service deployment is performed at each frame while request dispatch is executed at each slot, each request dispatch decision is generated when the decision variables of service deployment are known. Then, since only the request dispatch variables \mathbf{y} (real number variables) is included, the request dispatch subproblem (problem (2)) is a linear programming problem and can be solved in linear time. As can be seen, solving the NP-hard subproblem of service deployment is where the challenge lies. Therefore, we will seek efficient suboptimal service deployment algorithms with approximate guarantees.

$$\max : \sum_{l \in \mathcal{L}} \sum_{n \in \mathcal{N}} \lambda_{ln} \sum_{m \in \mathcal{N}} y_{lnm}, \quad (2a)$$

$$\text{s.t. : (1b), (1d), (1e),} \quad (2b)$$

$$y_{lnm} \leq I_{(l,m) \in \mathcal{S}}, \quad \forall l \in \mathcal{L}, n \in \mathcal{N}, m \in \mathcal{N}, \quad (2c)$$

$$y_{lnm} \geq 0, \quad \forall l \in \mathcal{L}, n \in \mathcal{N}, m \in \mathcal{N}. \quad (2d)$$

First, we reformulate the service deployment subproblem as a combinatorial optimization problem. Let $\mathcal{S} \subseteq \mathcal{L} \times \mathcal{N}$ denote the set consisting of selected individual service deployment elements (l, n) , where $(l, n) \in \mathcal{S}$ denotes deploying a replica of service l on edge node n , i.e., $x_{ln} = 1$. Then, we let $\Omega(\mathcal{S})$ denote the optimal objective value of problem (1) under fixed service deployment variables

$\mathbf{x} = (x_{ln})$, where $x_{ln} = 1$ when and only when $(l, n) \in \mathcal{S}$. Then, $\Omega(\mathcal{S})$ can be computed by solving the request dispatch subproblem (2). Finally, we can rewrite the service deployment subproblem as problem (3).

$$\max : \Omega(\mathcal{S}), \quad (3a)$$

$$\text{s.t. : } \sum_{l:(l,n) \in \mathcal{S}} r_l \leq R_n, \quad \forall n \in \mathcal{N}, \quad (3b)$$

$$\mathcal{S} \subseteq \mathcal{L} \times \mathcal{N}. \quad (3c)$$

Next, we design a joint scheduling algorithm based on submodular function optimization to solve the service deployment subproblem. The idea of SD-SFM is to iteratively select elements to add to the selected set \mathcal{S} of service deployments (see Algorithm 1, line 4) continuously in a greedy manner, while evaluating the benefits brought by such action (see Algorithm 1, line 7) and selecting the service deployment element that brings the most benefits to add to the set \mathcal{S} in each round of iterations.

Algorithm 1: Service Deployment based on Submodular Function Maximization (SD-SFM)

input : $r_l, k_l, \lambda_{ln}, T_l, O_l, R_n, K_n, t_{nm}$

output: Service deployment variables \mathbf{x} , request dispatch variables \mathbf{y}

```

1 Initialization: the service deployment set  $\mathcal{S} = \emptyset$ , the optimization value  $\omega = 0$ ;
2 repeat
3   Initialize temporary variables at a new round:  $\omega' \leftarrow 0, (l', n') = (0, 0)$ ;
4   for  $(l, n) \in (\mathcal{L} \times \mathcal{N}) \setminus \mathcal{S}$  do
5     if  $\mathcal{S} \cup \{(l, n)\}$  satisfies the constraints (3b) and (3c) then
6       Solve problem (2) to derive the objective function value
7          $\Omega(\mathcal{S} \cup \{(l, n)\})$ ;
8       if  $\Omega(\mathcal{S} \cup \{(l, n)\}) > \omega'$  then
9         Update the temporary optimization target value
10         $\omega' = \Omega(\mathcal{S} \cup \{(l, n)\})$ ;
11        Update temporary decision pair  $(l', n') = (l, n)$  for service
12        deployment;
13  if  $(l', n') \neq (0, 0)$  then
14    Update the optimal set of decisions for service deployment
15     $\mathcal{S} = \mathcal{S} \cup \{(l', n')\}$ ;
16 until  $(l', n') = (0, 0)$ ;
17 Convert  $\mathcal{S}$  into its vector representation  $\mathbf{x}$ ;
18 Solve problem (2) to derive the request dispatch decisions  $\mathbf{y}$  based on  $\mathbf{x}$ ;

```

In addition, we consider a more general scenario in which LC services and BE services can be deployed simultaneously within an edge node for smart grids. This is because although the LC service has a strict QoS requirement, its resource occupation is not always maintained at a high level due to the fluctuation of request arrival strength. The resource efficiency (or throughput) of the smart

grid cluster can be further improved if the BE services can make full use of the remaining resources of the LC services. Therefore, we design Algorithm 2 to optimize the hybrid scheduling of LC and BE services. We denote the set of frames as $\mathcal{F} = \{1, \dots, f, \dots, F\}$, and for each frame f , its corresponding set of slots as $\mathcal{T}_f = \{1, \dots, t_f, \dots, T_f\}$.

Algorithm 2: Two-tier hybrid scheduling algorithm

```

1 for  $f \in \mathcal{F}$  do
2   Execute the algorithm SD-SFM for the deployment of LC services;
3   Calculate  $R'_n = R_n - \sum_{l \in \mathcal{L}} x_{ln} \cdot r_l$  for each node  $n \in \mathcal{N}$ ;
4   Execute the algorithm SD-SFM on remaining storage resources for the
   deployment of BE services;
5   for  $t_f \in \mathcal{T}_f$  do
6     Execute request dispatch for LC services by solving problem (2);
7     Calculate  $K'_m = K_m - \sum_{l \in \mathcal{L}} k_l \sum_{n \in \mathcal{N}} \lambda_{ln} \cdot y_{tmn}$  for each node  $m \in \mathcal{N}$ ;
8     Execute request dispatch on remaining computing resources for BE
     services by solving problem (2);

```

4.2 Effectiveness Analysis

To prove that the SD-SFM algorithm is conditionally near-optimal with approximation guarantees, we need to show that the objective function of problem (3) is a monotone submodular function under certain conditions.

Lemma 1. *The objective function $\Omega(\mathcal{S})$ of problem (3) is a monotonic submodular function if the following conditions hold, and the SD-SFM algorithm can conditionally generate near-optimal decisions: a) $\lfloor R_n/r_l \rfloor \leq 1, \forall n \in \mathcal{N}, \forall l \in \mathcal{L}$; or b) $\sum_{l \in \mathcal{L}} k_l \sum_{n \in \mathcal{N}} \lambda_{ln} \leq K_n, \forall n \in \mathcal{N}$.*

Proof. First, we prove the monotonicity of the objective function: Since $\mathcal{S}_1 (\mathcal{S}_1 \subseteq \mathcal{S}_2 \subseteq \mathcal{L} \times \mathcal{N})$ is a subset of \mathcal{S}_2 , \mathcal{S}_2 relaxes the constraint of condition (2c) compared to \mathcal{S}_1 , so the objective function value grows, i.e., $\Omega(\mathcal{S}_1) \leq \Omega(\mathcal{S}_2)$.

Second, we prove the submodularity of the objective function [4]. Since there is no competition for resources among requests under the condition of Lemma 1, the throughput brought by (l_1, n_1) in the service deployment decision $\mathcal{S}_1 \cup \{(l_1, n_1)\}$ is $\Omega(\mathcal{S}_1 \cup \{(l_1, n_1)\}) - \Omega(\mathcal{S}_1)$. Consider the following case, first deploy the service in $\mathcal{S}_1 \cup \{(l_1, n_1)\}$ and then deploy the service in $\mathcal{S}_2 \setminus \mathcal{S}_1$, then the requests originally served by the replica (l_1, n_1) may be offloaded to the replica in $\mathcal{S}_2 \setminus \mathcal{S}_1$, so the number of requests served by (l_1, n_1) may become less, i.e., $\Omega(\mathcal{S}_1 \cup \{(l_1, n_1)\}) - \Omega(\mathcal{S}_1) \geq \Omega(\mathcal{S}_2 \cup \{(l_1, n_1)\}) - \Omega(\mathcal{S}_2)$, where $\mathcal{S}_1 \subseteq \mathcal{S}_2 \subseteq \mathcal{L} \times \mathcal{N}, ((l_1, n_1) \in (\mathcal{L} \times \mathcal{N}) \setminus \mathcal{S}_2)$.

Finally, it's obvious that $(\mathcal{L} \times \mathcal{N}, \mathcal{J})$ is an independence system [5] where $\mathcal{J} \subseteq 2^{\mathcal{L} \times \mathcal{N}}$ is the set of all feasible solutions to problem (3). We consider any two feasible service deployment sets $\mathcal{S}_1 \subseteq \mathcal{S}_2$ and \mathcal{S}_2 . As it can reduce the number

of replicas by at most p times when transforming the set \mathcal{S}_1 into \mathcal{S}_2 , constraints (3b) and (3c) constitute a p -independence system, where $p = \left\lceil \frac{\max r_l}{\min_{l:r_l>0} r_l} \right\rceil$.

To maximize the monotone submodular function subject to p -independence system constraints, the greedy algorithm has an approximation rate of $1/(p+1)$ [4]. Therefore, the SD-SFM algorithm has $1/(p+1)$ approximation rate with problem (1), which proves the theoretical feasibility of the SD-SFM algorithm.

5 Performance Analysis

In our experiments, we simulate service requests based on the Alibaba cluster dataset [2], and then forward these requests to the smart grid cluster randomly (an edge cluster consists of $|\mathcal{N}|=6$ edge nodes for smart grids and $|\mathcal{L}| \in [10, 20]$ types of services). We scale the attributes in the dataset, e.g., the “plan_mem” field and “plan_cpu” field in the “pai_task_table” dataset are used as the storage resources r_l and computing resources k_l required by the request, the “end_time” field minus the “start_time” field is used as the processing time O_l of a service, “cap_mem” field and “cap_cpu” field in the “pai_machine_spec” dataset are used as the storage capacity R_n and processing capacity K_n of a node, respectively. Specifically, r_l, k_l, R_n, K_n are scaled to the data ranges $[0 : 5, 1]; [0 : 5, 1]; [1, 8]; [4, 32]$, respectively. Besides, the values for transmission delay between edge nodes t_{nm} are drawn from $(0, |\mathcal{N}|/2]$, where $t_{nn} = 0$. Meanwhile, the maximum response time T_l tolerated [6] by the LC service is the sum of O_l and a random value drawn from $(0, |\mathcal{N}|/2]$, and that by the BE service is the sum of O_l and $\max(t_{nm})$. The request arrival rate of a node, $\lambda_n = \sum_{l \in \mathcal{L}} \lambda_{ln}$, is drawn from $[5, 50]$, while the value of λ_{ln} is based on its popularity in the dataset. In addition, a time frame contains $|T_f| \in [10, 60]$ time slots in the experiments.

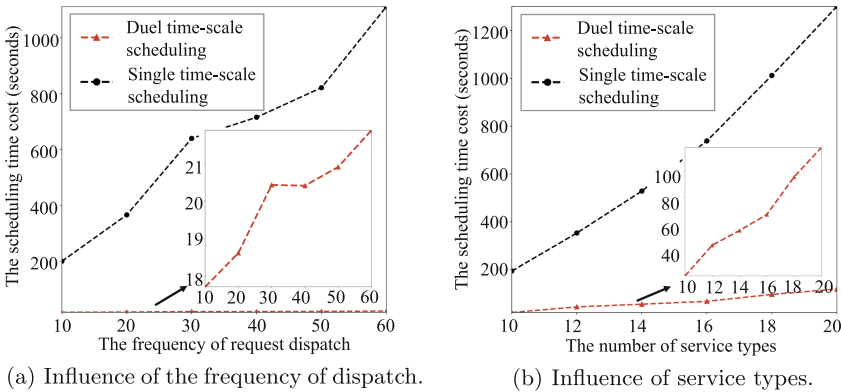


Fig. 2. Time cost comparison of dual/single time-scale scheduling.

To verify the beneficial effect of Algorithm 1 (i.e., SD-SFM algorithm) under the dual time-scale scheduling design, we compared the scheduling time cost

under single time-scale (executing one service deployment before each request dispatch) with that under dual time-scale. As shown in Fig. 2(a), the scheduling time cost keeps increasing as the frequency of request dispatch increases, but the cost of single time-scale one grows much faster than the dual time-scale one. It is worth noting that in the dual time-scale, the frequency of request dispatch is equal to the number of time slots in a frame. Figure 2(b), on the other hand, illustrates that the scheduling time cost increases with the number of service types as well, and the single time-scale one also grows much faster than the dual time-scale one. Therefore, the dual time-scale scheduling design will be more beneficial to reduce the scheduling time cost of the system compared to the single time scale.

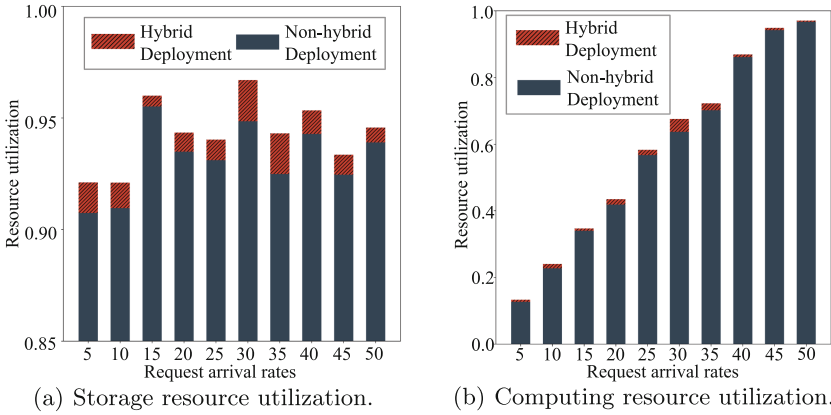


Fig. 3. Resource utilization of hybrid deployment versus non-hybrid deployment.

To verify the effectiveness of the SD-SFM algorithm in the two-tier hybrid scheduling algorithm, we conducted experiments under different request arrival rates. Figure 3(a) shows that the storage resource utilization under hybrid deployment of LC services and BE services is always higher than that without hybrid deployment of services at different request arrival rates. Figure 3(b) demonstrates that the computing resource utilization increases with the request arrival rate and is always higher under hybrid service deployment than non-hybrid service deployment. These experimental results validate the effectiveness of Algorithm 2 and also illustrate that hybrid deployment helps to improve resource utilization.

To further validate the effectiveness of the SD-SFM algorithm, we designed several different service deployment baseline strategies: 1) Service Deployment with Linear Programming Relaxation (SD-LPR): first relaxing problem (1) to a linear programming problem [12] to solve, and then rounding the service deployment variables to integer variables under the constraints of problem (1). 2) Service Deployment in Descending order of requests Demands for each Edge node

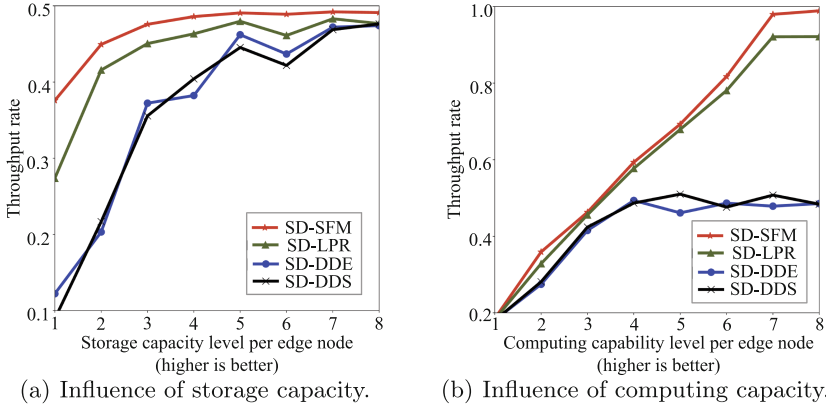


Fig. 4. The performance of SD-SFM against baselines with various resource settings.

(SD-DDE): considering each edge node $n \in \mathcal{N}$ in turn, deploying services on edge n in descending order (i.e., descending order of λ_{ln}) [13] according to the request arrival rate of each service l arriving at edge node n under the constraint of storage resources. 3) Service Deployment in Descending order of requests Demands for the whole edge-cloud cluster System (SD-DDS): Consider the total request arrival rate of each service l arriving within the whole edge-cloud cluster system (denoted as $\Lambda_l = \sum_{n \in \mathcal{N}} \lambda_{ln}$), and then deploy the services on each edge node in descending order of Λ_l [13] subject to the storage resource constraint. From Fig. 4(a), it can be seen that the throughput rates under different service deployment strategies improve as the storage resource capacity of the edge nodes increases, but the throughput rates under the SD-SFM algorithm are always higher than the other baseline strategies. Similarly, Fig. 4(b) demonstrates that the throughput rates under different service deployment strategies are increasing with the increase of computing resource capacity of the edge nodes, and the SD-SFM algorithm always brings higher throughput rates than other baseline strategies. It can be seen that the SD-SFM algorithm has superiority over other baseline strategies.

To summarize, although our experimental setup is more stringent than the conditions in Lemma 1, the theoretical near-optimality of Algorithm 1 is guaranteed if the constraints (1c) and (1d) in our scenario are relaxed. In fact, the experimental results do verify the superior performance of the algorithm.

6 Conclusion

First, this paper studies the scheduling optimization problem of edge-cloud smart grid clusters under the consideration of both system utility and QoS guarantees. By proving that the problem is NP-hard, this paper decouples two subproblems, such as service deployment and request dispatch, and gives feasible and efficient solutions to each of them. Secondly, this paper designs the heuristic

algorithm SD-SFM according to the monotonicity and submodularity of the service deployment subproblem and analyzes the near-optimality of the algorithm SD-SFM. Thirdly, this paper employs dual time-scale scheduling to coordinate two subproblems for reducing scheduling overhead and proposes a coordination algorithm for hybrid deployment of LC services and BE services within a resource channel to improve the utilization of resources of smart grid clusters. Finally, this paper verifies the feasibility as well as the efficiency of the proposed algorithm using experiments driven by real data sets. In future work, we will consider the overall joint scheduling optimization problem of resource slicing, service deployment and request dispatch to further improve resource efficiency.

Acknowledgement. This work is supported by the Science and Technology Project of State Grid Corporation of China under Grant 5700-202130263A-0-0-00.

References

1. Chen, S., Delimitrou, C., et al.: Parties: QoS-aware resource partitioning for multiple interactive services. In: International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), pp. 107–120 (2019)
2. Chen, W., Ye, K., Wang, Y., Xu, G., Xu, C.Z.: How does the workload look like in production cloud? analysis and clustering of workloads on Alibaba cluster trace. In: IEEE International Conference on Parallel and Distributed Systems (ICPADS), pp. 102–109 (2018)
3. Farhadi, V., Mehmeti, F., He, T., Porta, T.L., et al.: Service placement and request scheduling for data-intensive applications in edge clouds. In: IEEE Conference on Computer Communications (INFOCOM), pp. 1279–1287 (2019)
4. Fisher, M.L., Nemhauser, G.L., Wolsey, L.A.: An analysis of approximations for maximizing submodular set functions-II. In: Balinski, M.L., Hoffman, A.J. (eds.) *Polyhedral Combinatorics*, pp. 73–87. Springer, Cham (1978). <https://doi.org/10.1007/BFb0121195>
5. Gupta, A., Roth, A., Schoenebeck, G., Talwar, K.: Constrained non-monotone submodular maximization: offline and secretary algorithms. In: International Workshop on Internet and Network Economics, pp. 246–257 (2010)
6. Hudson, N., Khamfroush, H., Lucani, D.E.: QoS-aware placement of deep learning services on the edge with multiple service implementations. In: 2021 International Conference on Computer Communications and Networks (ICCCN), pp. 1–8 (2021)
7. Liang, Y., Ge, J., Zhang, S., Wu, J., Pan, L., Zhang, T., et al.: Interaction-oriented service entity placement in edge computing. *IEEE Trans. Mobile Comput.* **20**(3), 1064–1075 (2021)
8. Liu, Q., Han, T., Moges, E.: EdgeSlice: slicing wireless edge computing network with decentralized deep reinforcement learning. In: IEEE International Conference on Distributed Computing Systems (ICDCS), pp. 234–244 (2020)
9. Luo, Q., Hu, S., Li, C., Li, G., Shi, W.: Resource scheduling in edge computing: a survey. *IEEE Commun. Surv. Tutorials* **23**(4), 2131–2165 (2021)
10. Nair, V., Bartunov, S., Gimeno, F., von Glehn, I., Lichocki, P., et al.: Solving mixed integer programs using neural networks. arXiv preprint [arXiv:2012.13349](https://arxiv.org/abs/2012.13349) (2020)
11. Ning, Z., Dong, P., Wang, X., Wang, S., Hu, X., Guo, S., et al.: Distributed and dynamic service placement in pervasive edge computing networks. *IEEE Trans. Parallel Distrib. Syst.* **32**(6), 1277–1292 (2021)

12. Poularakis, K., Llorca, J., Tulino, A.M., Taylor, I., Tassiulas, L.: Joint service placement and request routing in multi-cell mobile edge computing networks. In: IEEE Conference on Computer Communications (INFOCOM), pp. 10–18 (2019)
13. Vlachou, A., Doulkeridis, C., Kotidis, Y., Norvag, K.: Monochromatic and bichromatic reverse top-k queries. *IEEE Trans. Knowl. Data Eng.* **23**, 1215–1229 (2011)
14. Wang, X., Han, Y., Leung, V.C., Niyato, D., Yan, X., Chen, X.: Convergence of edge computing and deep learning: a comprehensive survey. *IEEE Commun. Surv. Tutorials* **22**(2), 869–904 (2020)



Deep Reinforcement Learning Based Computation Offloading in Heterogeneous MEC Assisted by Ground Vehicles and Unmanned Aerial Vehicles

Hang He^{1,4}, Tao Ren^{1,4}(✉), Meng Cui², Dong Liu³, and Jianwei Niu^{1,4}

¹ Hangzhou Innovation Institute, Beihang University, Hangzhou 310051, China
{hehang, rentaoBH, niujianwei}@buaa.edu.cn

² CNPC Engineering Technology R and D Company Limited, Beijing, China
cuimengdri@cnpc.com.cn

³ College of Computer and Information Engineering, Henan Normal University,
Xinxiang 453007, China

⁴ School of Computer Science and Engineering, Beihang University, Beijing 100191, China

Abstract. Compared with traditional mobile edge computing (MEC), heterogeneous MEC (H-MEC), which is assisted by ground vehicles (GVs) and unmanned aerial vehicles (UAVs) simultaneously, is attracting more and more attention from both academia and industry. By deploying base stations (along with edge servers) on GV or UAVs, H-MEC is more suitable for access-demand dynamically-changing network environments, e.g., sports matches, traffic management, and emergency rescue. However, it is non-trivial to perform real-time user association and resource allocation in large-scale H-MEC environments. Motivated by this, we propose a shared multi-agent proximal policy optimization (SMAPPO) algorithm based on the centralized training and distributed execution framework. Due to the NP-hard difficulty of jointly optimizing user association and resource allocation for H-MEC, we adopt the actor-critic-based online-policy gradient (PG) algorithm to obtain near-optimal solutions with low scheduling complexities. In addition, considering the low sampling efficiency of PG, we introduce proximal policy optimization to increase the training efficiency by importance sampling. Moreover, we leverage the idea of centralized training and distributed execution to improve the training efficiency and reduce scheduling complexity, so that each mobile device makes decisions based only on local observation and learns other MDs' experience from a shared replay buffer. Extensive simulation results demonstrate that SMAPPO can achieve more satisfactory performances than traditional algorithms.

Keywords: Mobile edge computing · Computation offloading · Deep reinforcement learning · Proximal policy optimization · Unmanned aerial vehicle

1 Introduction

The explosive growth of real-time computation-intensive mobile applications, such as online gaming, automatic driving, and virtual reality (VR), has strict requirements of

low latency delivery by mobile devices (MDs) [1]. Nonetheless, MDs are usually constrained by finite battery capacity and computing resources and are not able to meet such requirements of the applications [2]. Fortunately, benefitting from the emergence of 5G, mobile edge computing (MEC) is proposed and envisioned as a promising technology to tackle the challenge by offloading computing-intensive and delay-sensitive tasks to the edge servers of nearby base stations [3].

In traditional MEC, edge servers are deployed in fixed terrestrial infrastructures, which not only leads to high deployment costs, but also is inadaptable for serving access-demand dynamically changing scenarios, e.g., sports matches, traffic management, and emergency rescue [4]. Therefore, heterogeneous MEC (H-MEC) [5], whose edge servers could be deployed at both terrestrial infrastructures, and movable ground vehicles (GVs) or unmanned aerial vehicles (UAVs), is considered as an effective complement of traditional MEC [6]. Owing to the high flexibility and convenience of deploying GV and UAVs, H-MEC is able to accommodate dynamic network environments with hotspot areas or emergency rescue activities on demand [7]. However, the high flexibility and dynamically-changing demand also bring intractable challenges for H-MEC, which include in-time decision-making, large-scale user association, and resource allocation under stringent scheduling constraints.

In existing works, [8] adopted the convex optimization (CO) method and heuristic search algorithm to solve task offloading and resource allocation for multi-server MEC networks. [9] proposed a coordinate descent method to maximize the computation rate for wireless-powered MEC. Although closed-form solutions could be obtained in the above CO-based methods, they require a considerable amount of iterations to reach an optimal solution and thus are not suitable for making real-time offloading decisions [10] in dynamically changing environments. [11] proposed a DNN-based framework to serve the dynamic H-MEC environment through incremental online learning [12], and [13] adopted a DRL-based framework to adjust the policy of DRL online in hybrid MEC networks. However, these existing learning-based methods are mostly inefficient for data sampling and suffer from slow convergence.

In view of the above issues, we propose a shared multi-agent proximal policy optimization (SMAPPO) algorithm to perform real-time online computation offloading and resource management for H-MEC, which also accommodates high efficiency in data-sampling and model-training. The main contributions of this article are generalized as follows:

- We optimize the online computation offloading and resource management solution in H-MEC by achieving the minimum system cost with Markov decision process, which is further worked out by using online multi-agent deep reinforcement learning algorithms with affordable scheduling complexities for each MD.
- We further improve the sampling efficiency by introducing importance sampling for the learning algorithm and increase the training efficiency by replacing long-term cumulative rewards with creditable reward advantages. Specifically, MD can learn from the experiences of other MDs through a shared replay buffer.
- Finally, we demonstrate the effectiveness and efficiency of our algorithm via extensive experiments. The simulation results demonstrate that SMAPPO can achieve more satisfactory performances than traditional algorithms do.

2 System Model

2.1 Network Model

As illustrated in Fig. 1, we consider a heterogeneous MEC network consisting multiple MDs and edge nodes including several UAVs and GVs. The sets of GVs and UAVs are denoted by $\mathcal{V} = \{1, 2, \dots, V\}$ and $\mathcal{U} = \{1, 2, \dots, U\}$, where V and U refer to the number of GVs and UAVs, respectively. Each GV/UAV is equipped with an edge server to serve MDs' computation-intensive and delay-sensitive applications.

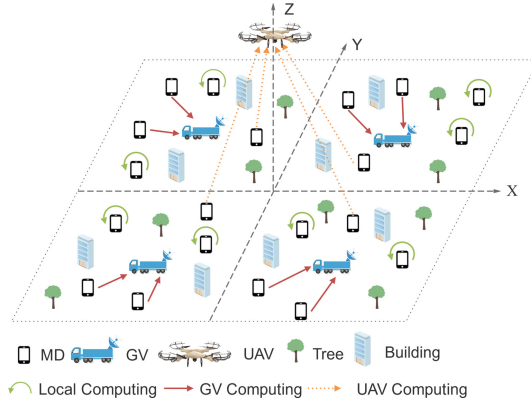


Fig. 1. H-MEC System model

We assume that there are M MDs randomly distributed in the ground and denote the set of MDs by $\mathcal{M} = \{1, 2, \dots, M\}$. Similar to [14], the whole running period is uniformly distributed as N equal parts and define the set of time slots by $\mathcal{N} = \{1, 2, \dots, N\}$. MD i at time slot n randomly generates a task J_i^n as follows:

$$J_i^n = \{D_i^n, F_i^n, T_i^{n,\max}\}, \quad (1)$$

where D_i^n represents the input data size, F_i^n is the required amount of CPU frequencies to compute one bit of task J_i^n , $T_i^{n,\max}$ denotes the maximum tolerable latency of task.

In this article, we adopt a complete offload method, where the task can be entirely processed either on the MD or on a GV/UAV. We denote o_i^n as the computation offloading decision of MD i at slot n as follows:

$$o_i^n = \begin{cases} 0 & \text{if Local Computing} \\ 1 & \text{if GV Edge Computing} \\ 2 & \text{if UAV Edge Computing} \end{cases}, \quad (2)$$

2.2 Communication Model

In this subsection, we give the channel gain models between GVs and MDs as well as between UAVs and MDs, respectively.

1) *MD-GV Channel Model*: Similar to [15], the channel gain between MD i and GV j can be expressed by:

$$g_{i,j}^n = 128.1 + 37.6 \cdot \log_{10}(d_{i,j}^n), \tag{3}$$

where $d_{i,j}^n$ is the Euclidean distance between MD i and GV j at time slot n .

Therefore, the offloading transmission rate between MD i and GV j at time slot n can be given by:

$$r_{i,j}^{n,V} = W_j \cdot \log_2 \left[1 + p_i^n g_{i,j}^n / \left(\sigma^2 + \sum_{u,u \neq i}^M p_u^n g_{u,j}^n \right) \right], \tag{4}$$

where p_i^n represents the transmission power of MD i at time slot n , and W_j is the channel bandwidth between MD i and GV j .

2) *MD-UAV Channel Model*: Similar to [16], the channel gain between MD i and UAV k can be calculated by

$$g_{i,k}^n = 20 \cdot \log(d_{i,k}^n) + 98.4 + \zeta_{LOS} \cdot P_{i,k}^{n,LOS} + \zeta_{NLOS} \cdot (1 - P_{i,k}^{n,LOS}), \tag{5}$$

where ζ_{LOS} and ζ_{NLOS} represent the additional loss due to line-of-sight and no-line-of-sight links, respectively. $d_{i,k}^n$ is the Euclidean distance between MD i and UAV k at time slot n .

Therefore, the transmission rate between MD i and UAV k at time slot n can be given by:

$$r_{i,k}^{n,U} = W_k \cdot \log_2 \left[1 + p_i^n g_{i,k}^n / \left(\sigma^2 + \sum_{u,u \neq k}^M p_u^n g_{u,k}^n \right) \right]. \tag{6}$$

where p_i^n expresses the transmitting power of MD i , and W_k is the channel bandwidth between MD i and UAV k .

2.3 Computation Model

1) *Local Computing*: If $o_i^n = 0$, the arrived task is processed by MD i . So, the latency of executing task in the local device is given by

$$\bar{t}_i^{n,L} = D_i^n \cdot F_i^n / f_i^{n,L}, \tag{7}$$

where $f_i^{n,L}$ is the local computing capacities of MD i . Accordingly, the energy consumption of processing task can be calculated by

$$\bar{e}_i^{n,L} = \kappa \cdot (f_i^{n,L})^{\xi-1} \cdot D_i^n \cdot F_i^n, \tag{8}$$

where κ express the switch capacitance coefficient of the chip architecture. With (7) and (8), the weighted cost of executing task locally is given by.

$$c_i^{n,L} = w_i^{t,L} \cdot \tilde{t}_i^{n,L} + w_i^{e,L} \cdot \bar{e}_i^{n,L}, \quad (9)$$

where $w_i^{t,L}$ and $w_i^{e,L}$ denote the weights of delay and energy consumption cost respectively, and $0 \leq w_i^{t,L} \leq 1, 0 \leq w_i^{e,L} \leq 1, w_i^{t,L} + w_i^{e,L} = 1$.

- 2) *GV Edge Computing*: If $o_i^n = 1$, the task of MD i is handled at the ground vehicle. Therefore, the latency of transmitting task remotely to ground vehicle is given by

$$\tilde{t}_{i,j}^{n,V} = D_i^n / r_{i,j}^n, \quad (10)$$

Accordingly, the energy consumption of transmitting task is expressed as

$$\bar{e}_{i,j}^{n,V} = p_i^n \cdot \tilde{t}_{i,j}^{n,V}, \quad (11)$$

where p_i^n expresses the transmitting power of MD i at time slot n . In addition, the latency of executing task remotely in ground vehicle is given by

$$\tilde{t}_{i,j}^{n,V} = D_i^n \cdot F_i^n / f_{i,j}^{n,V}. \quad (12)$$

Accordingly, the energy consumption of executing task is expressed as

$$\bar{e}_{i,j}^{n,V} = z_j^n \cdot \tilde{t}_{i,j}^{n,V}. \quad (13)$$

With (10), (11), (12) and (13), in GV edge computing, the total cost of MD i and ground vehicle is given as follow:

$$c_i^{n,V} = w_i^{t,V} \cdot \left(\tilde{t}_{i,j}^{n,V} + \bar{t}_{i,j}^{n,V} \right) + w_i^{e,V} \cdot \left(\bar{e}_{i,j}^{n,V} + \bar{e}_{i,j}^{n,V} \right). \quad (14)$$

where $w_i^{t,V}$ and $w_i^{e,V}$ denote the weights of delay and energy consumption cost respectively, and $0 \leq w_i^{t,V} \leq 1, 0 \leq w_i^{e,V} \leq 1, w_i^{t,V} + w_i^{e,V} = 1$.

- 3) *UAV Edge Computing*: If $o_i^n = 2$, the task of MD i is processed at the UAV. Therefore, the latency of transmitting task remotely in UAV is given by

$$\tilde{t}_{i,k}^U = D_i / r_{i,k}^U, \quad (15)$$

Accordingly, the energy consumption of transmitting task is expressed as

$$\bar{e}_{i,k}^U = p_i^n \cdot \tilde{t}_{i,k}^U, \quad (16)$$

Besides, the latency of executing task remotely in UAV is given by

$$\tilde{t}_{i,k}^U = D_i^n \cdot F_i^n / f_{i,k}^{n,U}, \quad (17)$$

where $f_{i,k}^{n,U}$ is the computing capacities assigned by the UAV k to MD i .

Accordingly, the energy consumption of executing task is expressed as

$$\bar{e}_{i,k}^U = z_k^n \cdot \tilde{t}_{i,k}^U. \quad (18)$$

With (15), (16), (17) and (18), in UAV edge computing, the total cost of MD i and unmanned aerial vehicle is given as follow:

$$c_i^{n,U} = w_i^{t,U} \cdot (\bar{t}_{i,k}^{n,U} + \bar{t}_{i,k}^{n,U}) + w_i^{e,U} \cdot (\bar{e}_{i,k}^{n,U} + \bar{e}_{i,k}^{n,U}), \quad (19)$$

where $w_i^{t,U}$ and $w_i^{e,U}$ denote the weights of delay and energy consumption cost respectively, and $0 \leq w_i^{t,U} \leq 1, 0 \leq w_i^{e,U} \leq 1, w_i^{t,U} + w_i^{e,U} = 1$.

3 Problem Formulation

In this chapter, we describe the computation offloading and resource management optimization problem in the H-MEC system.

Given the above, the system cost of MD i at time slice n is given by

$$C_i^n = \begin{cases} C_i^{n,L} & \text{if } o_i^n = 0; \\ C_i^{n,V} & \text{if } o_i^n = 1; \\ C_i^{n,U} & \text{if } o_i^n = 2. \end{cases} \quad (20)$$

To this end, the average system cost in H-MEC throughout period is expressed by

$$C_{sys} = \frac{1}{M} \cdot \frac{1}{N} \cdot \sum_{i=1}^M \sum_{n=1}^N C_i^n. \quad (21)$$

To sum up, the average system cost minimization problem under corresponding system constraints is formulated as below.

$$\begin{aligned} \mathcal{P} : & \min_{(o_i^n, p_i^n, f_{i,j}^{n,L}, f_{i,j}^{n,V}, f_{i,k}^{n,U})} C_{sys} \\ \text{st. } & \forall i \in \mathcal{M}, j \in \mathcal{V}, k \in \mathcal{U}, n \in \mathcal{N} \\ C1 : & o_i^n \in \{0, 1, 2\}; C2 : p_i^n \leq p_i^{L,max}; C3 : f_i^{n,L} \leq f_i^{L,max} \\ C4 : & \sum_{i=1}^M f_{i,j}^{n,V} \leq f_j^{V,max}; C5 : \sum_{i=1}^M f_{i,k}^{n,U} \leq f_k^{U,max}; C6 : \bar{t}_i^{n,L} \leq T_i^{n,max} \\ C7 : & \bar{t}_i^{n,V} + \bar{t}_{i,j}^{n,V} \leq T_i^{n,max}; C8 : \bar{t}_i^{n,U} + \bar{t}_{i,j}^{n,U} \leq T_i^{n,max} \\ C9 : & \sum_{n=1}^t \bar{e}_i^{n,L} \leq E_i^{L,max}; C10 : \sum_{n=1}^t \sum_{i=1}^M (\bar{e}_{i,j}^{n,V} + \bar{e}_{i,j}^{n,V}) \leq E_j^{V,max}, \forall t \in \mathcal{N} \\ C11 : & \sum_{n=1}^t \sum_{i=1}^M (\bar{e}_{i,k}^{n,U} + \bar{e}_{i,j}^{n,U}) \leq E_k^{U,max}, \forall t \in \mathcal{N} \end{aligned} \quad (22)$$

4 Algorithm

In the section, its first part is devoted to the introduction of the reinforcement learning. Then, we will present the process of transforming the optimization problem into MDP [17] problem. Finally, we demonstrate in detail our proposed SMAPPO algorithm.

4.1 Preliminary Work

- 1) *State*. The state of each MD includes its task details, channel state, and battery information. Thus, the state of MD i at time step n can be expressed as

$$s_i^n = \left(D_i^n, F_i^n, T_i^{n,max}, E_i^{n,now}, g_{i,j}^{n,V}, g_{i,k}^{n,U} \right), \quad (23)$$

where $E_i^{n,now}$ depicts the current remaining capacity of MD i at time step n

- 2) *Action*. The action of each MD includes its offloading indicator, transmitting power, and allocated computing capacity. At the time slot n , the action of MD i can be represented by

$$a_i^n = \left(o_i^n, p_i^n, f_i^{n,L}, f_{i,j}^{n,V}, f_{i,k}^{n,U} \right). \quad (24)$$

- 3) *Reward*. We aim to search the optimal computation offloading and resource management solution to minimize the average system cost under system constraints in H-MEC system. So, we define the reward r_i^n of MD i at time slot n as

$$r_i^n = R(s_i^n, a_i^n) = -C_i^n. \quad (25)$$

4.2 Problem Transformation

Based on above definition of the three elements of reinforcement learning, the trajectory of MD i during N time slot can be represented as:

$$\tau_i = \left\{ s_i^1, a_i^1, s_i^2, a_i^2, \dots, s_i^N, a_i^N \right\}. \quad (26)$$

Accordingly, the probability and total reward of the trajectory τ_i are calculated as follow:

$$R(\tau_i) = \sum_{n=1}^N r_i^n, \quad (27)$$

and

$$p_\theta(\tau_i) = p(s_i^1) \prod_{n=1}^N \left[p_\theta(a_i^n | s_i^n) \cdot p(s_i^{n+1} | s_i^n, a_i^n) \right], \quad (28)$$

respectively, where θ is the network parameter of the Actor. The average reward is denoted as

$$\bar{R}_\theta = \sum_{\tau} R(\tau) P_\theta(\tau) = E_{\tau \sim p_\theta(\tau)} [R(\tau)]. \quad (29)$$

To this end, the problem \mathcal{P} could be transformed into object function $\mathcal{P}1$ as below:

$$\mathcal{P}1 : \max_{\theta} \bar{R}_\theta. \quad (30)$$

4.3 Algorithm Implementation

To address problem $\mathcal{P}1$, we propose a shared multi-agent proximal policy optimization (SMAPPO) algorithm based on the centralized training and distributed execution (CTDE) framework, as shown in Fig. 2. From the flow diagram, the whole framework can be divided into three parts: decentralized execution, data collection and centralized training. Specifically, each MD first interacts with the H-MEC environment based on its observation of the local state, generating a batch of learning experiences. Then, the learned experiences are employed to train a shared policy and value function for all MDs, with adoption of the generalized advantage estimation and importance sampling method. Last, each MD shares the trained policy and continuously interacts with the environment.

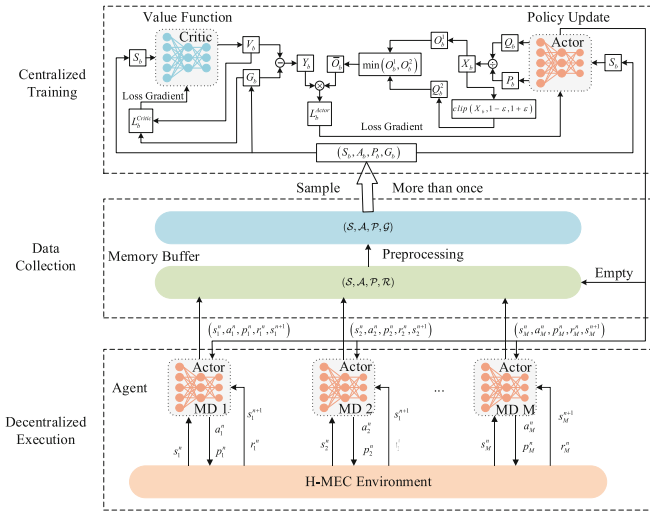


Fig. 2. The distributed execution and centralized training framework.

In the decentralized execution phrase, each MD i obtains its initial state s_i^n from the H-MEC environment, gives the probability distribution of actions p_i^n and the sampled action value a_i^n according to its Actor network, and attains the reward r_i^n and next state s_i^{n+1} from the environment. When a time slot ends, MD insert an experience tuple $(s_i^n, a_i^n, p_i^n, r_i^n, s_i^{n+1})$ into the shared replay buffer. At the end of an episode, we preprocess the experience tuples (S, A, P, R) . In the data collection phase, experience tuples are processed to convert immediate reward into cumulative reward. The experience tuple changes from (S, A, P, R) to (S, A, P, G) .

During the centralized training phrase, we randomly sample a lots of experiences tuples (S_b, A_b, P_b, G_b) from the replay buffer to calculate the loss function L_b^{Actor} and L_b^{Critic} of Actor network and Critic network, and update the network parameters of two network. Due of the importance sampling, we can sample and update multiple times. After the training, we copy the updated network parameters of Actor network to all MDs.

We provide the pseudocode of the SMAPPO algorithm in Algorithm 1.

Algorithm 1 SMAPPO Algorithm

```

1: Initiate the Actor  $\pi$  and Critic  $V$  with parameter  $\theta' \leftarrow \theta$  and  $\phi' \leftarrow \phi$ ; Initiate the memory;
2: for episode  $e = 1, 2, \dots, E$  do
3:   for time step  $n = 1, 2, \dots, N$  do
4:     for MD  $i = 1, 2, \dots, M$  do
5:       Interactive with the environment, and store tuple  $(s_i^n, a_i^n, p_i^n, r_i^n, s_i^{n+1})$  into memory.
6:     end for
7:   end for
8:   for update times  $t = 1, 2, \dots, T$  do
9:     for sample times  $s = 1, 2, \dots, S$  do
10:      Select randomly  $B$  mini-batch size tuples;
11:      Compute advantages using (32); Computing Actor loss and Critic loss using (34) and (36);
12:      Apply gradient descent  $\nabla \theta$  and  $\nabla \phi$  by Adam Optimizer;
13:      Update the Actor  $\pi$  and Critic  $V$  with  $\theta' \leftarrow \theta$  and  $\phi' \leftarrow \phi$ ;
14:    end for
15:  end for
16: end for
    
```

The gradient of object function is given by:

$$\begin{aligned}
 \nabla \bar{R}_\theta &= \sum_{\tau} R(\tau) \cdot \nabla p_\theta(\tau) = \sum_{\tau} R(\tau) \cdot p_\theta(\tau) \cdot \nabla \log p_\theta(\tau) \\
 &= E_{\tau \sim p_\theta(\tau)} [R(\tau) \cdot \nabla \log p_\theta(\tau)] \approx \frac{1}{B} \sum_{b=1}^B [R(\tau_b) \cdot \nabla \log p_\theta(\tau_b)] \\
 &= \frac{1}{B} \sum_{b=1}^B \sum_{n=1}^N [R(\tau_b) \cdot \nabla \log p_\theta(a_b^n | s_b^n)],
 \end{aligned} \tag{31}$$

where B is the mini-batch size of each sampling. In order to add a baseline and assign suitable credit, we introduce advantage function [18] to replace total reward. The definition of advantage function denotes as follow:

$$G_\theta(s_b^n, a_b^n) = \sum_{n'=n}^N (\gamma^{n'-n} \cdot r_b^{n'} - V_\phi(s_b^n)), \tag{32}$$

where γ is the discount factor of future reward [19]. Therefore, the gradient $\nabla \bar{R}_\theta$ can change as:

$$\nabla \bar{R}_\theta = \frac{1}{B} \sum_{b=1}^B \sum_{n=1}^N [G_\theta(s_b^n, a_b^n) \cdot \nabla \log p_\theta(a_b^n | s_b^n)]. \tag{33}$$

To improve the efficiency of data sampling, we choose off-policy [20] to replace on-policy, the loss function of the Actor network is expressed by:

$$\begin{aligned}
 J_{\theta'}(\theta) &= E_{(s_t, a_t) \pi_{\theta'}} \left[\frac{p_{\theta}(s_t, a_t)}{p_{\theta'}(s_t, a_t)} G_{\theta'}(s_t, a_t) \right] \\
 &\approx \sum_{(s_t, a_t)} \min(X_t \cdot G_{\theta'}(s_t, a_t), \text{clip}(X_t, 1 - \varepsilon, 1 + \varepsilon) \cdot G_{\theta'}(s_t, a_t)),
 \end{aligned} \tag{34}$$

where θ' is the network parameters of Actor of each MD, θ is the network parameters that we need to train. X_t can be calculated as follow:

$$X_t = \frac{p_{\theta}(s_t, a_t)}{p_{\theta'}(s_t, a_t)}. \tag{35}$$

Besides, the loss function of the Critic can be represented as:

$$J_{\phi} = \sum_{(s_t, a_t)} (G_{\theta}(s_t, a_t) - V_{\phi}(s_t))^2. \tag{36}$$

5 Simulation Results

In this chapter, we first give some simulation settings which are followed by the simulation results and their discussion and analysis.

5.1 Simulation Setting

In the simulation, we assume that four GVs and one UAV in a $800 \times 800 m^2$ square zone, with the coordinate of UAV as $(0, 0, 50)$. The four GVs are located at $(200, 200)$, $(-200, 200)$, $(-200, -200)$, and $(200, -200)$, respectively. Similar to [21], The bandwidth set 30 MHz as default value, the available computing capability of MD, GV and UVA is 1 GHz, 30GHz and 60GHz, respectively. Each MD is randomly moved with moving velocity of 5 m/s. The input data size of generated task and the amounts of computing cycles of finish on bit are sampled from [400000, 50000] bits and [800, 900] cycles/s. The maximum tolerance delay is uniformly sampled from [0.8, 0.9] s. Both the actor and the critic network have four hidden layers $1024 \times 512 \times 256 \times 128$. Use ReLU as activation function and define the 10^{-3} and 3×10^{-3} as learning rate of the actor and critic, respectively. The buffer capacity of memory is 8000, and the batch size of each sampling is 64. Let the discount factor of future reward as 0.99 and define the parameter clip as 0.2.

5.2 Performance Evaluation

The performance of SMAPPO is compared with the below benchmark algorithms:

- (1) LOCAL: all tasks are processed within the MDs;

- (2) Random: half of the tasks are executed locally and the other half are moved randomly to edge nodes;
- (3) MADQN: Consider that the action space is discrete, we use multi-agent deep Q network algorithm to generate offloading decision. The task offloading is optimized by the MADQN algorithm.

We first compare the performance of SMAPPO with those of the other benchmarks w.r.t the average reward with different numbers of MDs, as shown in Fig. 3. It can be noticed that with the rise of numbers of MDs, the average reward of SMAPPO, DQN and RANDOM gradually decline. Thus, we can reasonably speculate the reason that with the increase of the numbers of MDs, more MDs choose to offload the task to edge nodes, which can cause the transmission time and energy consumption to rise.

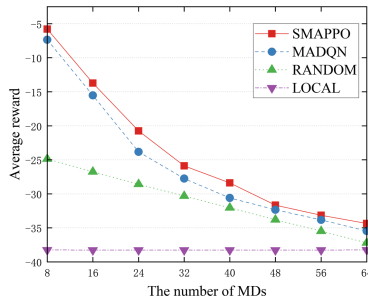


Fig. 3. Performance comparison with different numbers of MDs.

For simplicity, we assume that the number of MD is 24 in subsequent experiment.

Then, we investigate the performance of SMAPPO with those of baselines w.r.t the average reward with different bandwidth of GVs, as illustrated in Fig. 4 (a). It could be noticed that with the rise of bandwidth of GVs, the average reward of SMAPPO and MADQN notably increase. The observation may base on the fact that the increase of bandwidth of GVs can cut down the transmission time and energy consumption. Next, we compare the performance of SMAPPO with those of benchmarks w.r.t the average reward with various computing capacity of GVs, as exhibited in Fig. 4(b). It could be noticed that with the rise of computing capacity of GVs, the average reward of SMAPPO and MADQN first increase notably then gently. We can safely draw that as the computing capacity of GVs increases, the execution time and energy consumption reduce.

Besides, the performance of SMAPPO is compared with those baselines w.r.t the average reward with different bandwidth of UAVs, as depicted in Fig. 5(a). It could be noticed that with the rise of bandwidth of UAVs, the average reward of SMAPPO and MADQN quickly increase. The phenomenon means that when the bandwidth of UAVs expands, the transmission time and energy consumption is cut down. Last, we illustrate the performance of SMAPPO with those baselines w.r.t the average reward with various computing capacity of UAVs, as shown in Fig. 5(b). It could be noticed that with the rise of computing capacity of UAVs, the average rewards of SMAPPO and MADQN first

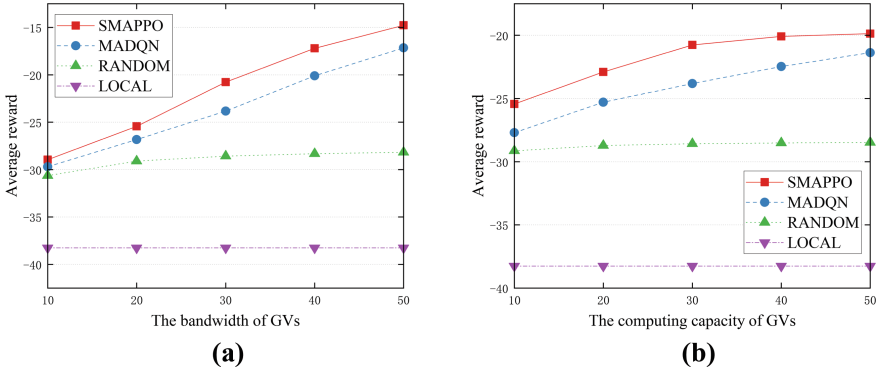


Fig. 4. Performance comparison with different GVs. (a). different bandwidths of GVs. (b). different computing capacities of GVs.

rise quickly then slow down. We can conclude that as the computing capacity of UAVs increases, time and energy consumption can be decreased.

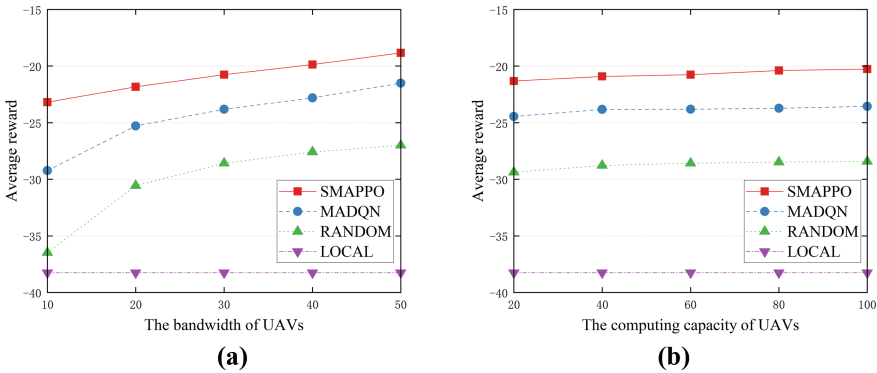


Fig. 5. Performance comparison with different UAVs. (a). different bandwidths of UAVs. (b). different computing capacities of UAVs.

6 Conclusion

In this article, we study the computation offloading and resource management problem in the H-MEC network with multiple mobile devices and edge nodes. Firstly, we build a heterogeneous MEC network consisting multiple MDs and edge nodes including several UAVs and GVs. And on this basis, we minimize the average system cost by searching the optimal offloading decision, transmitting power, and allocated computing capacity solution. Then, we transform the optimization problem into a Markov Decision Process (MDP) based on related knowledge of reinforcement learning. Next, we propose a shared multi-agent PPO (SMAPPO) based on deep reinforcement learning for a dynamic system. We further improve the sampling efficiency by introducing importance sampling for

the learning algorithm and increase the training efficiency by replacing long-term cumulative rewards with creditable reward advantages. On the one hand, MD can learn from the experiences of other MDs through a shared replay buffer. The training efficiency of the model can be improved by learning the experience of others MDs. On the other hand, we also improve the sampling efficiency of data through importance sampling. By sharing experience and network parameter, our proposed algorithm outperforms traditional methods in performance. Meanwhile, it also avoids the dimensionality disaster that centralized scheduling may face. Finally, we demonstrate the effectiveness and efficiency of SMAPPO via extensive experiments.

References

1. Li, J., Gao, H., Lv, T., et al.: Deep reinforcement learning based computation offloading and resource allocation for MEC. In: 2018 IEEE Wireless Communications and Networking Conference (WCNC). IEEE, pp. 1–6 (2018)
2. Bi, S., Ho, C.K., Zhang, R.: Wireless powered communication: opportunities and challenges. *IEEE Commun. Mag.* **53**(4), 117–125 (2015)
3. Sabella, D., Vaillant, A., Kuure, P., et al.: Mobile-edge computing architecture: the role of MEC in the Internet of Things. *IEEE Consumer Electronics Magazine* **5**(4), 84–91 (2016)
4. Wang, Y., Fang, W., Ding, Y., et al.: Computation offloading optimization for UAV-assisted mobile edge computing: a deep deterministic policy gradient approach. *Wireless Netw.* **27**(4), 2991–3006 (2021)
5. Zheng, X., Cai, Z.: Privacy-preserved data sharing towards multiple parties in industrial IoTs. *IEEE J. Sel. Areas Commun. (JSAC)* **38**(5), 968–979 (2020)
6. Zhang, W., Li, L., Zhang, N., et al.: Air-ground integrated mobile edge networks: a survey. *IEEE Access* **8**, 125998–126018 (2020)
7. Cai, Z., Zheng, X.: A private and efficient mechanism for data uploading in smart cyber-physical systems. *IEEE Trans. Network Sci. Eng. (TNSE)* **7**(2), 766–775 (2020)
8. Tran, T.X., Pompili, D.: Joint task offloading and resource allocation for multi-server mobile-edge computing networks. *IEEE Trans. Vehicular Technol.* **68**(1), 856868 (2018)
9. Bi, S., Zhang, Y.J.: Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading. *IEEE Trans. Wireless Commun.* **17**(6), 4177–4190 (2018)
10. Cai, Z., Xiong, Z., Xu, H., Wang, P., Li, W., Pan, Y.: Generative adversarial networks: a survey towards private and secure applications. *ACM Comput. Surv.* Accepted
11. Jiang, F., Wang, K., Dong, L., et al.: AI driven heterogeneous MEC system with UAV assistance for dynamic environment: challenges and solutions. *IEEE Network* **35**(1), 400408 (2020)
12. Liang, Y., Cai, Z., Yu, J., Han, Q., Li, Y.: Deep learning based inference of private information using embedded sensors in smart devices. *IEEE Network Mag.* **32**(4), 8–14 (2018)
13. Jiang, F., Wang, K., Dong, L., et al.: Deep-learning-based joint resource scheduling algorithms for hybrid MEC networks. *IEEE Internet of Things Journal* **7**(7), 62526265 (2019)
14. Du, Y., Wang, K., Yang, K., et al.: Energy-efficient resource allocation in UAV based MEC system for IoT devices. In: 2018 IEEE Global Communications Conference (GLOBECOM). IEEE, pp. 1–6 (2018)
15. Radio network planning and optimisation for UMTS. John Wiley & Sons (2006)
16. Holis, J., Pechac, P.: Elevation dependent shadowing model for mobile communications via high altitude platforms in built-up areas. *IEEE Trans. Antennas Propag.* **56**(4), 1078–1084 (2008)

17. Guo, D., Tang, L., Zhang, X., et al.: Joint optimization of handover control and power allocation based on multi-agent deep reinforcement learning. *IEEE Trans. Veh. Technol.* **69**(11), 13124–13138 (2020)
18. Schulman, J., Moritz, P., Levine, S., et al.: High-Dimensional Continuous Control Using Generalized Advantage Estimation. arXiv preprint [arXiv:1506.02438](https://arxiv.org/abs/1506.02438) (2015)
19. Niu, J., Song, W., Shu, L., Atiquzzaman, M.: Bandwidth-adaptive application partitioning for execution time and energy optimization. In: *IEEE International Conference on Communications*, art. no. 6655122, pp. 3660–3665 (2013)
20. Niu, J., Song, E.W., Atiquzzaman, M.: Bandwidth-adaptive partitioning for distributed execution optimization of mobile applications. *J. Network Computer Applications* **37**(1), 334–347 (2014)
21. Peng, H., Shen, X.: Multi-agent reinforcement learning based resource management in MEC- and UAV-assisted vehicular networks. *IEEE J. Sel. Areas Commun.* **39**(1), 131–141 (2020)



Synchronous Federated Learning Latency Optimization Based on Model Splitting

Chen Fang^{1,3}, Lei Shi^{1,3}(✉), Yi Shi², Jing Xu^{1,3}, and Xu Ding^{1,3}

¹ School of Computer Science and Information Engineering, Hefei University of Technology, Hefei 230009, China

shilei@hfut.edu.cn

² Department of ECE, Virginia Tech, Blacksburg, VA 24061, USA

³ Engineering Research Center of Safety Critical Industrial Measurement and Control Technology, Ministry of Education, Hefei 230009, China

Abstract. Federated Learning (FL) is a distributed machine learning approach which is suitable for edge computing environment. While in this environment, how to take full advantage of the computing resources on end devices and edge servers is still a difficult problem. Especially for the synchronous federated learning, computing resources among different participants will lead to extra time cost and cause resource waste. In this paper, we try to reduce the time cost and the computing resource waste by using model splitting and task scheduling. We first establish the mathematical model and find it can not be solved directly. Then we design our algorithm which we name as the Federated Learning Offloading Acceleration (FLOA) algorithm to obtain a sub-optimal solution. The FLOA algorithm first uses the Partition Points Selection method to reduce the size of the solution space, then proposes a task offloading method based on matching theory. Experiments and simulations show that compared to the other three calculation methods, the single iteration time is reduced by 47%, 28%, 14% under our algorithm in turn.

Keywords: Edge computing · Federated learning · Model splitting

1 Introduction

With the development and popularity of AI applications, it has become a trend for deploying AI applications on smart devices. The key to the deployment of AI applications is to use the rich data distributed on smart devices for training AI models. The data on smart devices contains a large amount of the user's privacy [1, 2]. Traditional cloud computing requires these data on devices to be transferred to the cloud centre, which will lead a large communication burden

The Work is Supported by Major Science and Technology Projects in Anhui Province (202003a05020009).

and threaten the data privacy. To solve these problems, some scholars are trying to combine edge computing and Federated Learning (FL) to train AI models [3].

Edge computing refers to the enabling technologies allowing computation to be performed at the edge of the network and is much efficient to process the data at the edge of the network [4]. For example, in [5] and [6], authors use computing resources from end devices and edge servers to process data. FL is a distributed machine learning approach suitable for edge computing [7]. In FL, participants collaborate with each other to train a shared DNN model together by using their own local data without sharing them [8,9]. In detail, each participant first trains a shared DNN model by using its local private data, and then uploads the model parameters to the parameter server for aggregation to obtain a global model. This process can be iterated several times until the trained model achieves the desired accuracy. FL has two different types of iterations, synchronous and asynchronous. The synchronous one means that model aggregation occurs after all participants complete local computation [10].

There are already some researches on combining edge computing with FL. In [11], a multi-layer federated learning protocol called HybridFL is designed for the Mobile Edge Computing (MEC) architecture, HybridFL improves the FL training process significantly in terms of shortening the federated round length, speeding up the global model's convergence and reducing end device energy consumption. [12] proposes a new FL-based client selection optimization to balance the trade-off between energy consumption of the edge clients and the learning accuracy of FL. [13] introduces a novel Hierarchical Federated Edge Learning (HFEL) framework, further formulate a joint computation and communication resource allocation and edge association problem for device users under HFEL framework to achieve global cost minimization.

In addition, model splitting can also accelerate the training process while protecting data privacy [14]. By using model splitting technique, a DNN model can be split inside between two successive layers into two partitions and then be deployed on different locations without losing accuracy [15]. [16] uses model splitting in FL to protect data privacy by placing the first layer of the DNN model on the end device, so there is no need to transmit sample data.

Previous work has made some contributions in combining edge computing and FL. However, they mostly focus on improvements to FL framework or aggregation protocols, and few researches use model splitting in federated learning to reduce latency. They do not take full advantage of the computing resources of end devices and edge servers in edge computing. In addition, they also ignore the time cost and resource waste in synchronous FL caused by the difference in computing resources of end devices. In this paper, we make full use of computing resources in edge computing environment through model splitting and task scheduling, which reduces the time cost of synchronous federated learning. First, we build a mathematical model for federated learning under end-to-edge collaborative edge computing. Edge computing scenario is complex and the mathematical model is difficult to solve directly because of the many variables. We first reduce the size of the solution space by filtering the split points, and Federated Learning

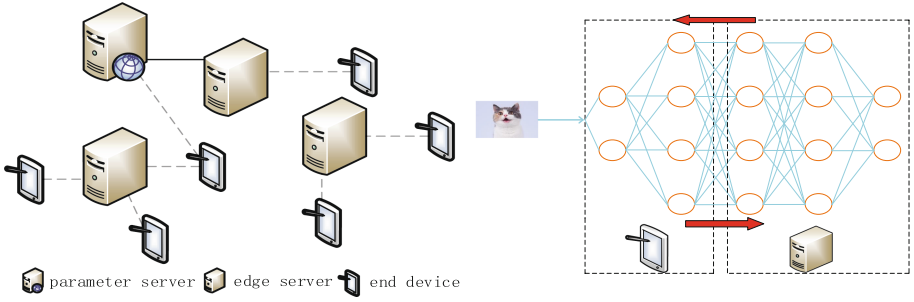


Fig. 1. System model.

Offloading Acceleration (FLOA) algorithm is designed based on matching theory to solve for the mathematical model.

The rest of this paper is organized as follows: In Sect. 2, we introduce system model and define our problem. In Sect. 3, we introduce the matching theory and the details of FLOA algorithm respectively. In Sect. 4, we give the simulation results and analyze them. In Sect. 5, we summarize this paper.

2 System Model and Problem Formulation

2.1 System Model

The system model is shown in Fig. 1. Suppose there are n end devices and m edge servers deployed in the scheduling network. Denote $e_i (i \in \{1 \dots n\}, e_i \in E)$ as one end device and $s_j (j \in \{1 \dots m\}, s_j \in S)$ as one edge server. Suppose all devices and servers are heterogeneous with different computing capabilities and suppose servers have better computing capabilities comparing with devices. Suppose there is a common DNN model needed to be trained in a distributed training manner by the whole network. Since we use model splitting technique, which means the DNN model will be split inside between two successive layers and then be trained separately on devices and servers. We also have a separate parameter server PS for updating parameters in the whole training process, devices and edge servers will send local training improvement to PS to obtain the global DNN model. The end devices are connected to edge servers and PS by wireless, the edge servers and PS are connected by wired.

Suppose we use the Stochastic Gradient Descent (SGD) algorithm to train the DNN model. Suppose the DNN model has v layers. Each device has its own data set, and the data set can be divided into many batches. For the training process, each data batch will undergo one forward propagation and one backward propagation. We call this process as an iteration. For a whole training process, each batch may be trained for several iterations and there are many iterations. But since we use the synchronous iteration method for training, which means iterations on different devices do not affect each other. So in our model, we only

consider one batch in each device for one iteration. Define an iteration for one batch on e_i as a task m_i . All tasks are computed in a serial manner on end devices and edge servers. For one task, an appropriate model splitting point will be selected according to the calculation amount, the size of parameter data and the size of intermediate results of each layer. Then the first half of the model will be trained on the end device, and the second half of the model will be trained on an edge server. For each task, the edge server will be selected according to the real-time status to undertake the training of the second half of the model. The total training time under synchronous training is the sum of all iterative training time. So we can minimize the total training time by minimizing the training time of each iteration.

Optimization objective: Minimize the training time of one iteration.

2.2 Problem Formulation

The time for completing task m_i can be expressed as

$$t_i = t_i^e + t_i^{trans} + t_i^s + t_i^w + t_i^{up}, \tag{1}$$

where t_i^e and t_i^s represent the training time for m_i on end device and edge server, respectively, t_i^w represents the waiting time for executing m_i on the edge server, t_i^{trans} represents the data transfer time between end device and edge server, and t_i^{up} represents the time to update the parameters on PS .

Define a binary variable x_i^r to indicate whether m_i is split at the r -th layer of DNN, we have

$$x_i^r = \begin{cases} 1 & :m_i \text{ is split at the } r\text{-th layer of the DNN;} \\ 0 & :\text{otherwise.} \end{cases} \tag{2}$$

Note that $\sum_{r=1}^v x_i^r = 1$, which means m_i should select one and only one model splitting point.

For the first item t_i^e , it consists of the forward propagation time $t_i^{e,f}$ and the backward propagation time $t_i^{e,b}$ on e_i . We have

$$t_i^e = t_i^{e,f} + t_i^{e,b} = \sum_{r=1}^v (x_i^r \cdot \sum_{w=1}^r (L_{e_i,w}^f + L_{e_i,w}^b)), \tag{3}$$

where $L_{e_i,w}^f$ and $L_{e_i,w}^b$ represent that for task m_i , the forward and the backward propagation time for the w -th layer when processing one batch on e_i .

For the second item t_i^{trans} , similar to t_i^e , it consists of the forward transmission time $t_i^{trans,f}$ and the backward transmission time $t_i^{trans,b}$ on e_i . We have

$$t_i^{trans} = t_i^{trans,f} + t_i^{trans,b} = 2 \cdot \sum_{r=1}^v x_i^r \cdot \frac{G_r}{B_{e,s}}, \tag{4}$$

where G_r represents the output size of the r -th layer in the forward phase, which is equal to the output data of the $(r+1)$ -th layer in the backward phase, and $B_{e,s}$

indicates the bandwidth between any device and any edge server (we suppose the bandwidths between any device and any edge server are the same).

For the third item t_i^s , denote a binary scheduling variable y_i^j to indicate whether m_i is executed on the server s_j . We have

$$y_i^j = \begin{cases} 1 & :m_i \text{ is executed on the server } s_j; \\ 0 & :\text{otherwise.} \end{cases} \quad (5)$$

Note that $\sum_{j=1}^m y_i^j = 1$, which means that a task can only select one edge server for offloading. Then the third item t_i^s can be expressed as

$$t_i^s = t_i^{s,f} + t_i^{s,b} = \sum_{j=1}^m (y_i^j \cdot \sum_{r=1}^v (x_i^r \cdot \sum_{w=r+1}^v (L_{s_j,w}^f + L_{s_j,w}^b))), \quad (6)$$

where $t_i^{s,f}$ and $t_i^{s,b}$ represent forward and backward propagation time on edge server s_j . $L_{s_j,w}^f$ and $L_{s_j,w}^b$ represent the m_i 's forward and the backward processing one batch time for layer w on s_j .

For the fourth item t_w , we first denote AT_i as the arrival time of task m_i to the edge server. We have

$$AT_i = t_i^{e,f} + t_i^{trans,f}, \quad (7)$$

where $t_i^{e,f}$ is the forward propagation time on the end device e_i , and $t_i^{trans,f}$ is the time that the intermediate result is transferred from the end device to the edge server.

When task m_i arrives, there may already have some tasks from other devices on the server forming a waiting queue. Therefore, the task should wait for the completion of these previous tasks. Then the fourth item t_i^w can be expressed as

$$t_i^w = \max\{0, I_i - AT_i\}, \quad (8)$$

where I_i represents the completion time of the task which precedes task m_i on the same task queue. I_i can be expressed as

$$I_i = \sum_{j=1}^m y_i^j \cdot \left(\sum_{i'=1}^n (y_{i'}^j \cdot Z_i(m_{i'}) \cdot (AT_{i'} + t_{i'}^w + t_{i'}^s)) \right), \quad (9)$$

where $Z_i(m_{i'})$ is a binary scheduling variable to indicate whether task $m_{i'}$ precedes task m_i . We have

$$Z_i(m_{i'}) = \begin{cases} 1 & :m_{i'} \text{ arrives one bit earlier than task } m_i; \\ 0 & :\text{otherwise.} \end{cases}$$

For the last item t_i^{up} , we have

$$t_i^{up} = \sum_{r=1}^v x_i^r \cdot \left(\sum_{w=1}^r \frac{d_w}{B_{e,ps}} \right), \quad (10)$$

where d_w is the parameter data size of the w -th layer, and $B_{e,ps}$ is the bandwidth between PS and any device (we suppose the bandwidths between any device and PS are the same).

Our optimization goal is to minimize the training time of one iteration. So we have

$$\begin{aligned} & \min(\max_{i \in \{1,2,\dots,n\}} t_i). \\ (1) \quad & (2) \quad (3) \quad (4) \quad (5) \quad (6) \quad (8) \quad (10) \\ & \sum_{r=1}^v x_i^r = 1; \\ & \sum_{j=1}^m y_i^j = 1. \end{aligned} \tag{11}$$

The analysis of the problem shows that there are two types of 0–1 variable in this optimization problem: x_i^r and y_i^j , while the others are constants. As each end device has its own decision variables, the number of variables is large. The two decision variables of one device in (6) are multiplied together, so this optimization problem is nonlinear. In summary, we learn that the optimization problem is very complex and difficult to solve directly.

3 Algorithm

In the last section, we give the whole problem model and find it is hard to be solved directly. In this section, we will try to solve it and give our algorithm. For solving it, we need to find some way to reduce the size of the solution space. If we use the model splitting technique, the DNN can be split at any layer. However, the characteristics of different layers, such as the size of the computation and the parameter data, vary greatly. So not all layers are suitable for splitting. In our algorithm, we will first reduce the number of split points between each device and each server with the help of the Partition Points Selection(PPS) algorithm in [17]. Then based on matching theory, we design our whole algorithm of selecting a split point and an offloaded server for each end device. We name our algorithm as the Federated Learning Offloading Acceleration (FLOA) algorithm.

3.1 Many-to-One Matching with Externalities

In this sub-section, we will give some definitions. These definitions will be useful for the FLOA algorithm description. The related two-sided matching problem is to assign agents of one set to agents of other disjoint set [18]. From Sect. 2 we know that one end device can only select one edge server, while one edge server can be assigned to multiple devices. So the server selection is many-to-one matching, which can be defined as follows.

Definition 1. Suppose e is one end device from E , and suppose s is one edge server from S . Define a many-to-one matching function μ on $E \cup S$, such that

- (1) $|\mu(e)| = 1$ for every device $e \in E$ and $|\mu(s)| \leq n$ for every server $s \in S$;
- (2) $e \in \mu(s)$ if and only if $s = \mu(e)$.

Based on Definition 1, we define $U_{e_i}(\mu)$ as the utility function for e_i on μ , and define $U_s(\mu)$ as the utility function for s on μ . The utility function can be used for measuring the matching effect. Then we have

$$U_{e_i}(\mu) = \frac{1}{t_i(\mu)}, \quad (12)$$

where $t_i(\mu)$ is the task training time of e_i under the matching state μ .

For $U_s(\mu)$, we have

$$U_s(\mu) = \frac{1}{\max_{i \in \{1, \dots, n\}} t_i(\mu)}, \quad (13)$$

Based on (12) and (13), we know that the matching effect is decided by the task training time $t_i(\mu)$. However, in our environment, training time for different tasks have high relationship, which means devices and edge servers care about more than their own matching. So traditional pairwise stable matching may not exist [19]. We continue to leverage the concept of two-sided exchange stability [20] on the following definition.

Definition 2. Define $\mu_e^{e'} = \{\mu \setminus \{(e, s), (e', s')\}\} \cup \{(e, s'), (e', s)\}$ as a swap matching, where $\mu(e) = s$, $\mu(e') = s'$, and $e \neq e'$.

A swap matching can enable device e and e' to swap their matched servers with each other, and remain the matching of other devices and servers unchanged. Notice that when e' is 0, it means that the edge server matched by device e is changed to s' , i.e. $\mu_e^0 = \{\mu \setminus \{(e, s)\}\} \cup \{(e, s')\}$, where $\mu(e) = s$ and $s \neq s'$.

Definition 3. Given a matching function μ and a pair of devices (e, e') , if there exists $\mu(e) = s$ and $\mu(e') = s'$, and satisfies: (1) $\forall x \in \{e, e', s, s'\}$, $U_x(\mu_e^{e'}) \geq U_x(\mu)$; (2) $\exists x \in \{e, e', s, s'\}$, $U_x(\mu_e^{e'}) > U_x(\mu)$; then we call (e, e') as a swap-blocking pair under μ .

Definition 4. A matching μ is said to be two-sided exchange stable if and only if there is no swap-blocking pairs in μ .

Definition 4 indicates that a matching is two-sided exchange stable if all devices fail to increase their own or the matching edge server's utility after changing the matching edge server.

3.2 FLOA Algorithm

Now we discuss the FLOA algorithm. The FLOA pseudo code can be found in Algorithm 1, and the detail steps are shown in the following.

Algorithm 1: Federated Learning Offloading Acceleration Algorithm

Input: E :Set of device; S :Set of server; \mathcal{P} :Set of split points to be selected.

Output: A two-sided exchange stable matching μ ; a split point strategy θ .

```

1 Initialization a matching  $\mu$ .
2 Initialize a split point strategy  $\theta$  for all devices based on the set  $\mathcal{P}$ .
3 repeat
4    $\exists e \in E, \mu(e) = s$ .
5   Re-select a split point in  $\mathcal{P}$  for device  $e$  such that  $U_e(\mu_e^0)$  is the
     maximum.
6   if device  $e$  meets  $U_e(\mu_e^0) \geq U_e(\mu)$  then
7     Device  $e$  sends an offload request to server  $s'$ .
8     if  $U_s(\mu_e^0) > U_s(\mu)$  or  $U_{s'}(\mu_e^0) > U_{s'}(\mu)$  then
9       The edge server  $s'$  receives the request:  $\mu \leftarrow \mu_e^0$ 
10      change split point strategy  $\theta$ .
11    else
12      The edge server  $s'$  rejects the request,
13      the split point strategy  $\theta$  remains unchanged.
14    $\exists e \in E, e' \in E, \mu(e) = s, \mu(e') = s',$  and  $e \neq e'$ .
15   Select a split point in  $\mathcal{P}$  for device  $e$  and  $e'$  such that  $U_e(\mu_e^{e'})$  and
      $U_{e'}(\mu_{e'}^{e'})$  is the maximum.
16   if device  $e$  and  $e'$  meet  $U_e(\mu_e^{e'}) \geq U_e(\mu)$  and  $U_{e'}(\mu_{e'}^{e'}) \geq U_{e'}(\mu)$  then
17     Device  $e$  sends an offload request to server  $s', e'$  sends an offload
       request to  $s$ .
18     if  $U_s(\mu_e^{e'}) > U_s(\mu)$  or  $U_{s'}(\mu_{e'}^{e'}) > U_{s'}(\mu)$  then
19       The edge servers  $s$  and  $s'$  both receive the request:  $\mu \leftarrow \mu_e^{e'}$ 
20       change split point strategy  $\theta$ .
21    else
22      The edge servers  $s$  and  $s'$  both reject the request,
23      the split point strategy  $\theta$  remains unchanged.
24 until Matching  $\mu$  meets Definition 4;
```

- step 1 Initialize a matching μ and a split point strategy θ based on the selected split point set \mathcal{P} , which is obtained from the PPS algorithm (line 1-2).
- step 2 Perform μ_e^0 for device e and re-select a split point in \mathcal{P} such that $U_e(\mu_e^0)$ is maximized under all available split points (line 4-5).
- step 3 If $U_e(\mu_e^0)$ is greater than $U_e(\mu)$, which is the utility of device e when matching the original matching server s , then let e send an offload request to s' (line 6-7).

- step 4 If $U_s(\mu_e^0)$ or $U_{s'}(\mu_e^0)$ increases after e changes the matched server, then let s' accept the offload request and update the matching μ and the split point policy θ , otherwise μ and θ remain unchanged. (line 6–13).
- step 5 Perform $\mu_e^{e'}$ for device e and e' , re-select a split point in \mathcal{P} such that $U_e(\mu_e^{e'})$ and $U_{e'}(\mu_e^{e'})$ is maximized under all available split points (line 14–15).
- step 6 If both e and e' have greater utility $U_e(\mu_e^{e'})$ and $U_{e'}(\mu_e^{e'})$ after swapping the matched servers than that before, then e and e' send offload requests to s' and s , respectively (line 16–17).
- step 7 If the utility of s or s' , $U_s(\mu_e^{e'})$ and $U_{s'}(\mu_e^{e'})$ increases after the swap, then s and s' accept the offload request and update μ and θ , otherwise μ and θ remain unchanged (line 18–23).
- step 8 Repeat above steps until Definition 4 is satisfied.

4 Simulation and Experiment

In this section, we demonstrate the validity of the previous sections of the work through simulation experiments. The DNN model to be trained is VGG16, and the training data is a set of RGB images of size $224 \times 224 \times 3$. All model training tasks are performed using the pytorch. The computing power of the end devices is simulated with the following 5 CPUs: AMD Ryzen 7 4800H, AMD Ryzen 9 3900X, i5-6200U, i5-11400H, and i7 11700F. And the computing power of the edge servers is simulated with the following GPU: NVIDIA GeForce RTX 2060(Note book).

First, we use CPUs and GPU to train the DNN model, and get the forward and backward propagation time for each layer of the DNN model on each end device and each edge server, i.e. the value of $L_{e_i,w}^f$, $L_{e_i,w}^b$, $L_{s_j,w}^f$ and $L_{s_j,w}^b$. Then we get the size of parameter data and intermediate data for each layer of the DNN model, i.e. the value of G_r and d_w . We assume that there are several end devices, 5 edge servers and one parameter server in the scheduling network, and that the value of bandwidth $B_{e,ps}$ is $6MBps$.

In Subsect. 4.1, according to PPS algorithm, we compare the total training time of the task when different layers are used as split point, and get the set of split points \mathcal{P} . In Subsect. 4.2, simulations are carried out with different numbers of end devices and different bandwidths between end devices and edge servers, respectively, to validate the proposed system model and algorithm.

4.1 Split Point Selection Experiment

First, we train the DNN model using a set of images of size $224 \times 224 \times 3$, and obtain the forward and backward propagation time for each layer of VGG16 on different end devices and different edge servers. Figure 2 shows the training time for each layer of VGG16 on one of the devices and one of the servers. We then study the structure of VGG16 to obtain the amount of data for the parameters and intermediate data for each layer of VGG16.

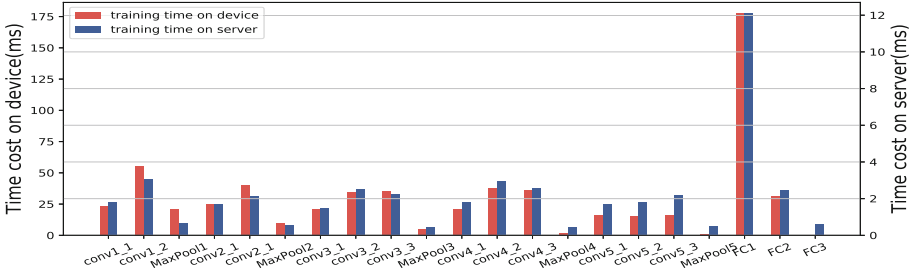


Fig. 2. The training time for each layer of VGG16 on device and server

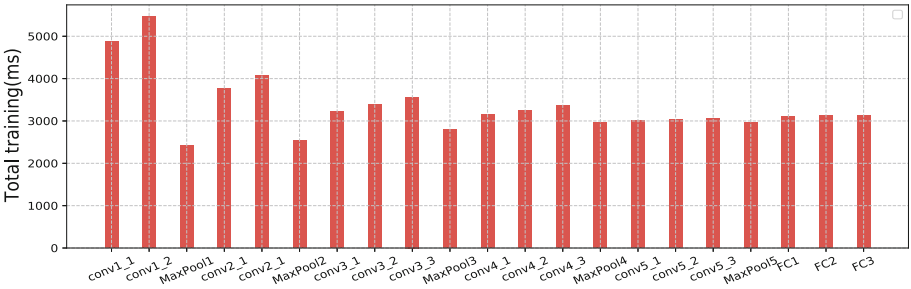


Fig. 3. The total training time under different split points of VGG16

We also set up a range of different bandwidths between the end devices and the edge servers. We then calculate the total training time ignoring waiting time for the task at different bandwidths with different layers as split point. Figure 3 shows that time when the bandwidth between the end device and the edge server is 6 Mbps. We obtain that the third, sixth and tenth layers are the three split points with the smallest total training time.

4.2 Simulation Results

In order to compare the effect of the algorithm with different number of devices and different bandwidths, first we set the bandwidth between the end device and the edge server to 6 Mbps, and the number of end devices from 40 to 90. Then we set bandwidth from 4 Mbps to 6 Mbps and fix the number of devices at 50. In both cases above, we calculate the time for a single iteration in different cases: Non-split, Fixed-split, Random strategy, and FLOA. The result is shown in Fig. 4.

In Fig. 4, FLOA indicates that we use the FLOA algorithm for training task scheduling. Non-split indicates that no model splitting is used, i.e. all training tasks are performed locally on the device. Fixed-split indicates that the split point is fixed and the training task randomly selects a edge server for offloading. Random strategy means randomly selecting a split point and a edge server for splitting and offloading.

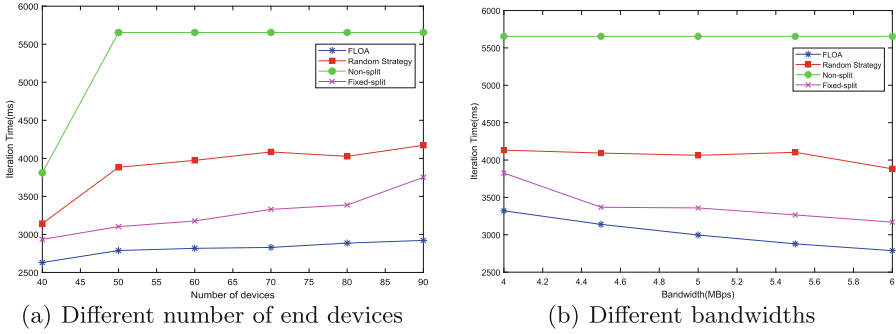


Fig. 4. Simulation result. (a) One iteration time with different number of devices. (b) One iteration time at different bandwidths.

As shown in the Fig. 4 (a), as can be seen from Non-split, the iteration time is always limited by the worst-performing device as the number of devices increases. FLOA obviously works better. Compared to Non-split, the one iteration time of FLOA is reduced by 47%. Compared to Fixed-split, the one iteration time is reduced by 14%. And compared to Random strategy, the one iteration time is reduced by 28%. In the Fig. 4 (b), FLOA is still obviously better than other solutions at different bandwidths.

5 Conclusion

In this paper, we use model splitting and task scheduling to reduce the overall training time for synchronous federated learning (FL) by reducing the time for one iteration. First we build mathematical models for synchronous federated learning in edge computing scenario. Then we analyse the mathematical model and design the corresponding solutions: using PPS algorithm to reduce the size of solution space and then proposing FLOA algorithm based on the two-sided matching theory to obtain the DNN splitting and offloading scheme. Experimental and simulation results show that using model splitting in synchronous FL, dynamically selecting split points and assigning training tasks can effectively reduce the time spent in synchronous FL.

References

1. Liang, Y., Cai, Z., Yu, J., Han, Q., Li, Y.: Deep learning based inference of private information using embedded sensors in smart devices. *IEEE Network* **32**(4), 8–14 (2018)
2. Cai, Z., Xiong, Z., Xu, H., Wang, P., Pan, Y.: Generative adversarial networks: a survey toward private and secure applications. *ACM Comput. Surv.* **54**(6), 1–38 (2021)
3. Ren, J., Yu, G., Ding, G.: Accelerating DNN training in wireless federated edge learning systems. *IEEE J. Sel. Areas Commun.* **39**(1), 219–232 (2021)

4. Shi, W., Cao, J., Zhang, Q., Li, Y., Xu, L.: Edge computing: vision and challenges. *IEEE Internet Things J.* **3**(5), 637–646 (2016)
5. Cai, Z., Shi, T.: Distributed query processing in the edge-assisted IoT data monitoring system. *IEEE Internet Things J.* **8**(16), 12679–12693 (2021)
6. Zhu, T., Shi, T., Li, J., Cai, Z., Zhou, X.: Task scheduling in deadline-aware mobile edge computing systems. *IEEE Internet Things J.* **6**(3), 4854–4866 (2019)
7. McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Singh, A., Zhu, J. (eds.) *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research*, vol. 54, pp. 1273–1282. PMLR, 20–22 April 2017
8. Pang, J., Huang, Y., Xie, Z., Han, Q., Cai, Z.: Realizing the heterogeneity: a self-organized federated learning framework for IoT. *IEEE Internet Things J.* **8**(5), 3088–3098 (2021)
9. Xiong, Z., Cai, Z., Takabi, D., Li, W.: Privacy threat and defense for federated learning with non-I.I.D. data in AIoT. *IEEE Trans. Ind. Inform.* **18**(2), 1310–1321 (2022)
10. Lim, W.Y.B., et al.: Federated learning in mobile edge networks: a comprehensive survey. *IEEE Commun. Surv. Tutorials* **22**(3), 2031–2063 (2020)
11. Wu, W., He, L., Lin, W., Mao, R.: Accelerating federated learning over reliability-agnostic clients in mobile edge computing systems. *IEEE Trans. Parallel Distrib. Syst.* **32**(7), 1539–1551 (2021)
12. Zheng, J., Li, K., Tovar, E., Guizani, M.: Federated learning for energy-balanced client selection in mobile edge computing. In: *2021 International Wireless Communications and Mobile Computing (IWCMC)*, pp. 1942–1947 (2021)
13. Luo, S., Chen, X., Wu, Q., Zhou, Z., Yu, S.: HFEL: joint edge association and resource allocation for cost-efficient hierarchical federated edge learning. *IEEE Trans. Wireless Commun.* **19**(10), 6535–6548 (2020)
14. Wang, X., Han, Y., Leung, V.C.M., Niyato, D., Yan, X., Chen, X.: Convergence of edge computing and deep learning: a comprehensive survey. *IEEE Commun. Surv. Tutorials* **22**(2), 869–904 (2020)
15. Zhou, Z., Chen, X., Li, E., Zeng, L., Luo, K., Zhang, J.: Edge intelligence: paving the last mile of artificial intelligence with edge computing. *Proc. IEEE* **107**(8), 1738–1762 (2019)
16. Qu, X., Hu, Q., Wang, S.: Privacy-preserving model training architecture for intelligent edge computing. *Comput. Commun.* **162**, 94–101 (2020)
17. Shi, L., Xu, Z., Shi, Y., Fan, Y., Ding, X., Sun, Y.: A DNN inference acceleration algorithm in heterogeneous edge computing: joint task allocation and model partition. In: Gao, H., Wang, X., Iqbal, M., Yin, Y., Yin, J., Gu, N. (eds.) *CollaborateCom 2020. LNICST*, vol. 349, pp. 237–254. Springer, Cham (2021). <https://doi.org/10.1007/978-3-030-67537-0-15>
18. Liu, Z., Wang, K., Zhou, M.T., Shao, Z., Yang, Y.: Distributed task scheduling in heterogeneous fog networks: a matching with externalities method. In: *2020 International Conference on Computing, Networking and Communications (ICNC)*, pp. 620–625 (2020)
19. Gale, D., Shapley, L.S.: College admissions and the stability of marriage. *Am. Math. Mon.* **120**(5), 386–391 (2013)
20. Bodine-Baron, E., Lee, C., Chong, A., Hassibi, B., Wierman, A.: Peer effects and stability in matching markets. In: Persiano, G. (ed.) *SAGT 2011. LNCS*, vol. 6982, pp. 117–129. Springer, Heidelberg (2011). <https://doi.org/10.1007/978-3-642-24829-0-12>



CodeDiff: A Malware Vulnerability Detection Tool Based on Binary File Similarity for Edge Computing Platform

Kang Wang^{1,2}, Longchuan Yan³, Zihao Chu⁴, Yonghe Guo³, Yongji Liu^{1(✉)}, Lei Cui¹, and Zhiyu Hao¹

¹ Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{wangkang, liuyongji, cuilei, haozhiyu}@iie.ac.cn

² School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

³ Information and Telecommunication Branch, State Grid, Beijing, China

⁴ State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China

Abstract. Malware detection has become a hot research pot as the development of Internet of Things and edge computing have grown in popularity. Specifically, various malware exploits firmware vulnerabilities on hardware platform, resulting in significant financial losses for both IoT users and edge platform providers. In this paper, we propose CodeDiff, a fresh approach for malware vulnerability detection on IoT and edge computing platforms based on the binary file similarity detection. CodeDiff is an unsupervised learning method that employs both semantic and structural information for binary diffing and does not require label data. Through the SkipGram with Negative Sampling, we generate the word vocabulary for instruction data. The Graph AutoEncoder is then used to embed both the semantic and structure information into the representation matrix for the CFG. After this, we employ the Improved Graph AutoEncoder to fuse all the function structures, function characteristics and function features to the fusion matrix. Finally, we propose the specific matrix comparison to achieve the high accuracy similarity results in short amount of time. Furthermore, we test the prototype on binary datasets OpenSSL and Curl. The results reveal that CodeDiff gives high performance on the binary file similarity detection, which contributes to identify malware vulnerability and improves the security of Internet of Things platforms.

1 Introduction

1.1 Motivation

Internet of Things combines all edge devices with the internet to improve urban life on multiple levels of modern smart city, enhance the productivity in industry 4.0 and power the construction of smart grid. However, with the widespread use of IoT and edge computing platforms, the safety for the firmware has become a critical issue. For example, in September 2017, a hacker discovered four attack vectors by analyzing the firmware of OFO shared bicycles and took control of the shared bicycles; in October 2017, LIFX smart light bulbs were also successfully hacked due to a password was leakage in its firmware. Two researchers from Northeastern University reversed the internal

firmware of Xiaomi IoT devices and found loopholes in the entire Xiaomi ecosystem at the ReCon BRX 2018 conference, as another example. A major aspect of IoT safety comes from that their malware has an amount size of numbers and distribution, but they usually have similar and even the same data sources, which makes them easy to get an attack through the same malware flaws. As a consequence, it is essential to analyze the binary file similarity for IoT and edge computing platform firmware to take precautions against malware vulnerability.

1.2 Limitations of Prior Art

IoT firmware security inspection is a cloud-based service for device manufacturers and operators, supply chain security management and service providers, and security evaluation and consulting agencies that provide non-invasive, multi-dimensional security risk inspection for device firmware. It could help manufacturers in swiftly identifying software security vulnerabilities and discovering numerous security risks such as weak passwords, certificate risks, privacy leakage, improper configuration, etc. Generally, it raises the security protection level of device firmware, for both the users and providers. There have been several proven techniques used in the firmware security detection for IoT device. CVE vulnerability based solutions [30] enables open source component CVE vulnerability detection and scans the Linux distribution package vulnerabilities. Code security [28] runs insecure library function usage detection and searches for the cracked or risky encryption algorithms. Sensitive information leakage [29] leverages but is not limited to the following methods: the source code leaks, information leaks in binary files, and plaintext storage of sensitive data in configuration files. However, they both rely on known malwares to detect the firmware vulnerabilities and lack of adaptation for new attacking methods. The main objective of this paper is to provide a novel firmware vulnerability detection tool for IoT and edge computing platform, which could track the vulnerability for even cross optimization/architecture/version binary files.

1.3 Proposed Approach

In this paper, we use similarity detection to identify the firmware vulnerabilities on the IoT and edge computing platform. Through the accurate and robust binary diffing, we can determine whether there exists the same source code vulnerability on the hardware platform and notify users and platform providers, thereby improving the safety of the IoT and edge computing platform. Specifically, we present the binary diffing system called CodeDiff, which can analyze binary file similarity across multiple optimization levels and CPU architecture, as well as the cross version revolution for the source code. Through extracting and summarizing the semantic features of the binary files, we use and extend the deep neural network-based method to carry out the feature extraction and similarity comparison of the binary files. We generate the representation matrix for each basic block by fusing the instruction and operand information. By transporting the semantic information and assembling basic block-level feature vectors, the graph is constructed through the control flow diagram inside the function. Combined with the structural information inside the function, the mixed feature embedded vector representation is obtained. Then we bring the function feature vector into the CFG as a node

feature and represent this as a network representation learning problem. Finally, we use the graph auto-encoder to obtain the low-dimensional vector representation of the binary file, and we carry out the similarity comparison process by comparing the vector cosine distance of the files. The entire file is represented as a fusion matrix in this manner, and we compare this fusion matrix between different binary files to determine the binary similarity.

1.4 Contribution

We provide CodeDiff, an unique malware vulnerability detection tool that detects vulnerabilities using binary file similarity detection to detect the vulnerabilities and improves the firmware safety for the IoT and edge computing platforms. CodeDiff uses an unsupervised model, the SkipGram and Graph AutoEncoder for binary diffing. In order to generate high-quality embeddings for basic blocks, we fuse the semantic information and the structure information through the Word2vec model, as well as the representation of the control flow graph. Meanwhile, we consider the structural information associated with functions and reduce the dimensionality of CFG to acquire the file feature vectors. Finally, we arrive at the optimal result for binary file similarity. Here we summarize the primary contribution as follows.

- We implement a prototype of CodeDiff. It first utilizes natural language processing (NLP) techniques to extract semantic information. Then, the SkipGram algorithm is executed to generate a feature vector for each basic block, which contains semantic and program-wide dependency information. Finally, an auto-encoder is used to fuse and compress the feature matrix for the binary file, as well as for the entire function call graph which represents the features of the function nodes.
- Extensive evaluations show that code diff can outperform state-of-the-art binary diffing tools for diffing across versions, across optimization levels and across architectures. A case study further demonstrates that CodeDiff can be analyzed to identify real-world vulnerabilities. This case study also evaluates that doing similarity analysis only for basic blocks and ignoring the entire file and function structure can easily get into a local dilemma.

2 Problem Definition

For two different binary files, CodeDiff precisely measures the similarity and characterizes the functional-level differences at a fine-grained level. The binary similarity difference problem is formalized as follows.

Definition: Given a collection of executables $F = \{F1, \dots, Fn\}$, and a query executable $Q = \{Q1, Q2, \dots, Qn\}$, containing several executable file sets of binary programs, we aim to determine for each executable $T_i \in F$ whether it has a similar relationship with $Q_i \in Q$.

Hypothesis: 1) Two input binaries are for the different architectures. Include x86, arm64, mips binaries as they are the most common binaries in the real world. 2) The binary files are not packaged but can be optimized by different compilers. Moreover, there will be distinct file-making code inputs generated even if the sources are the same. For packaged malware binaries, we assume that they could be opened and decompiled before getting submitted.

3 Method Overview

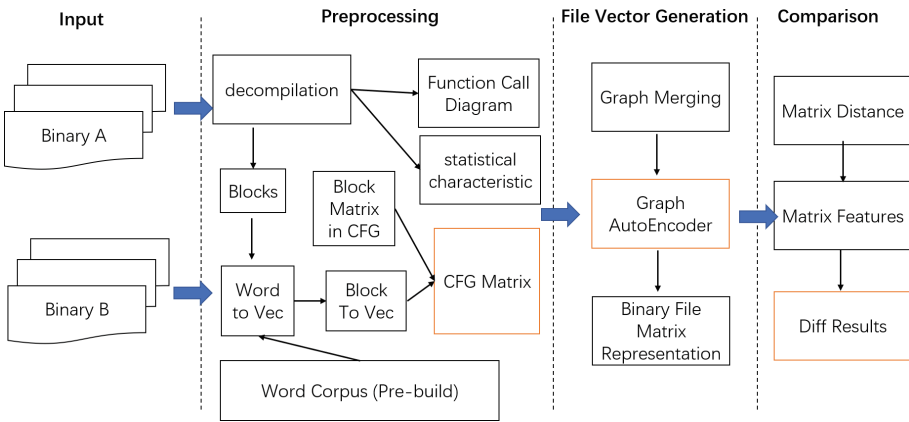


Fig. 1. CodeDiff: system architecture

The system takes the binary files as input and the similarity comparison result of embedded vectors as output. It performs feature processing on binary files in two dimensions to obtain the low-dimensional feature vectors. In order to get the feature vector of the basic block, CodeDiff first disassembles the binary file to get the command of the basic block. Then the embeddings are then generated by CodeDiff using the SkipGram method. Meanwhile, CodeDiff uses the control flow diagram inside the function to generate the feature vector of each function. Finally, CodeDiff extracts feature vectors for the entire file through the function call graph of the entire file, and CodeDiff compares the similarity of the files by the multi-layer perceptron.

The whole system consists of four main parts: 1) preprocessing; 2) embedding generation; 3) graph vector generation; 4) contrastive analysis between vectors. Preprocessing consists of two main parts: the generation of the control flow graph and the generation of the feature vector of functions.

After generation, the feature vector of the function is used as the node attribute, which belongs to the function call graph in the file. Then the nodes with attributes are processed by the graph embedding algorithm, reducing the dimension of the entire file on the basis of semantics and structure. The feature vectors of different files are compared to determine the similar relationship between different files.

4 Preprocessing

Preprocessing analyzes binary files and generates input for embedding generation. More specifically, it generates an Interprocedural Control Flow Graph (ICFG) for the binary file, then a Word2vec model is applied to generate an embedding for each command-line token (the instruction and operand). These generated token embeddings are further transformed into feature vectors which represents the characteristic of a basic block.

4.1 Disassemble

Since binary files can not be directly utilized for similarity detection, it is essential to choose an excellent disassembly tool for binary code disassembling. Currently, there are various mature disassembly tools available for researchers to choose from by need. There have been several typical disassembly tools, e.g., online disassembly tool ODA [20], commercial disassembly tool IDA Pro and Hopper, open-source disassembly tool Capstone [15]. These disassembly tools generally support features such as multiple instruction architecture sets, multiple platforms, and multiple compilation optimization options. We choose the IDA Pro disassembly tool which has the highest performance in those characteristics in terms of the stability, reliability, and functional diversity. The binary file is decompiled into assembly code by IDA Pro in this paper. In the disassembly process, the function is divided into basic blocks according to the jump of the instruction. Those basic blocks are formed as the nodes of the control flow chart, and the connection relationship of them, known as the jump position of the instruction, is generated as the flow chart edges. The control flow chart and the control flow graph are formed in this way, and the decompiled data is statistically integrated to obtain more detailed structural features inside the file.

4.2 Generation of Control Flow Graph

By combining the call graph with the control flow graph for each function, CodeDiff leverages IDA pro [14] to extract basic block information. In the meantime, IDA pro generates an interprocedural CFG (ICFG) which provides program-wide context information. This information contributes a lot to distinguishing semantically similar basic blocks in various settings.

4.3 Consideration of Structural Features

Besides the CFG of the function, we take into consideration of the structural features in the function as well as the statistical features of the numerical constants, which offers a high reference to judge the similarity of binary files. We count and calculate numerical features such as out-degree, in-degree, number of numerical constants, number of nodes and etc.

4.4 Generation of Semantic Features

To achieve high feature extraction performance, we take account of the semantic information, from the low-dimensional basic block to the high-dimensional binary file.

Specifically, we generate feature vectors for each basic block from the semantic information. The whole process includes two subtasks: token embedding generation and feature vector generation. Here we train a token embedding model derived from Word2Vec, then we make use of this model to generate the token (opcode and operand) embeddings. Following that, we generate the feature vector of the basic block from the token embedding. We explain the details of semantic feature generation in the following steps.

Normalization: Before the training of the token embedding model, we normalize the serialized code, which can decrease the difference in serialized code that comes from different compilation choices. In particular, we refine the code in the following normalization process. 1) All numeric constant values are replaced with the string “im”; 2) All general purpose registers are renamed according to their lengths; 3) Pointers are replaced with the string “prt”; 4) The function address is replaced with the string “addr”; 5) For the function to be jumped, the string “func” is replaced.

Model Training: CodeDiff treats the negative sampling as our modified Word2Vec version of the sentence algorithm [13]. Here we train a token embedding model that normalizes the negative sampling by learning token embeddings. This training step is only needs to be completed once then we can use this model to generate a vector for the token with no limitations. A word embedding is just a vector that is learned to capture the contextual semantics of the word meaning from a given article. There are multiple methods for generating vector representations of words, including the most popular models of Continuous Bag of Words (CBOW) and Skip-Gram [21], both of which utilize the model proposed by Mikolov et al. Here we take advantage of the Skip-Gram model to represent the target through context (Fig. 2).

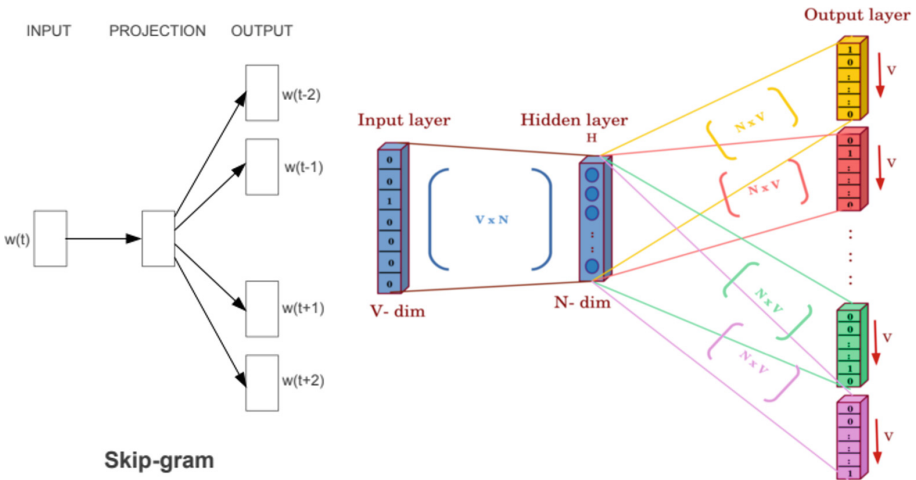


Fig. 2. SkipGram, model overview

SkipGram: Based on the token embeddings, the feature vectors of the basic blocks get generated. Since each basic block contains multiple instructions, each instruction involves an opcode and potentially multiple operands. We compute the average of the operand embeddings, and concatenate them with the opcode embeddings to generate instruction embeddings. Furthermore, we summarize the inner of the instruction block to formulate the block feature vector. In this paper, we improve the SkipGram through the negative sampling method, which is called the SkipGram of Negative Sampling (SGNS). We introduce the details in the following.

1) **Probability function:** $w(c, j)$ is the j -th word predicted on the c -th context position; $w(O, c)$ is the actual word present on the c -th context position; $w(I)$ is the only input word; and $u(c, j)$ is the j -th value in the U vector when predicting the word for c -th context position.

$$p(w_{c,j} = w_{O,c} | w_I) = \frac{\exp u_{c,j}}{\sum_{j'=1}^V \exp u_{j'}} \quad (4.4.a)$$

2) **Loss function:** as we want to maximize the probability of predicting $w(c, j)$ on the c -th context position, we can represent the loss function L as follows.

$$\mathcal{L} = -\log P(w_{c,1}, w_{c,2}, \dots, w_{c,C} | w_o) = -\sum_{c=1}^C u_{c,j^*} + \sum_{c=1}^C \log \sum_{j=1}^V \exp(u_{c,j}) \quad (4.4.b)$$

As a result, the fundamental block is standardized, and the semantic information is extracted by SkipGram. Furthermore, two loss layers, softmax and negative sampling are used in this algorithm. The weight gradient of the last layer is no longer related to the derivative of the activation function but is only proportional to the difference between the output value and the true value. The backpropagation is multiplicative, so the update of the entire weight matrix will be accelerated, which means that the convergence is faster at this time (Fig. 3).

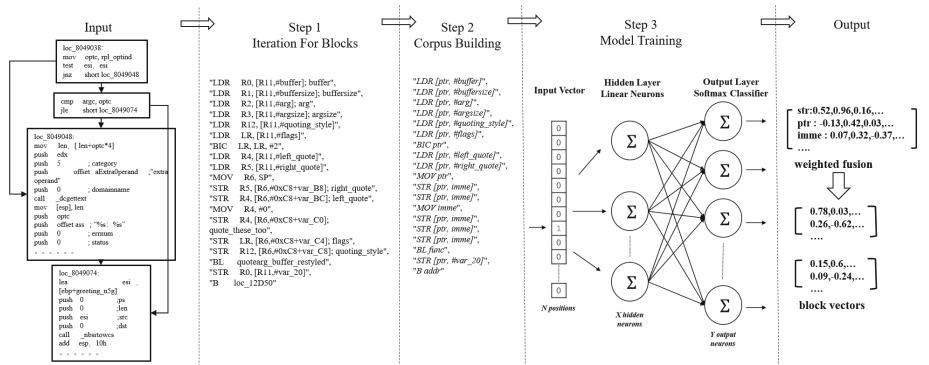


Fig. 3. Vector generation for basic blocks

5 Vector Generation for Binary Files

In previous steps, we generate the CFG and feature vectors, as well as the basic block embeddings. Specifically, CodeDiff performs node feature processing on the basic block and merges the feature vector of each CFG graph with the node block vector into one graph. Here we model this problem as a graph embedding problem. In our solution, we first make use of the Graph AutoEncoder (GAE) [23] to embed the feature vectors into the adjacent matrix, then compress this matrix to obtain a low-dimensional feature vector as the middle representation for the function. Second, we use another GAE to substitute this middle representation into the call function graph, resulting a low-dimensional feature vector to represent the binary file. In the following sections, we first describe the enhanced GAE algorithm in depth and the method of generating low-dimensional feature vectors. Then we demonstrate the necessity of graph merging and report how CodeDiff accomplishes it (Fig. 4).

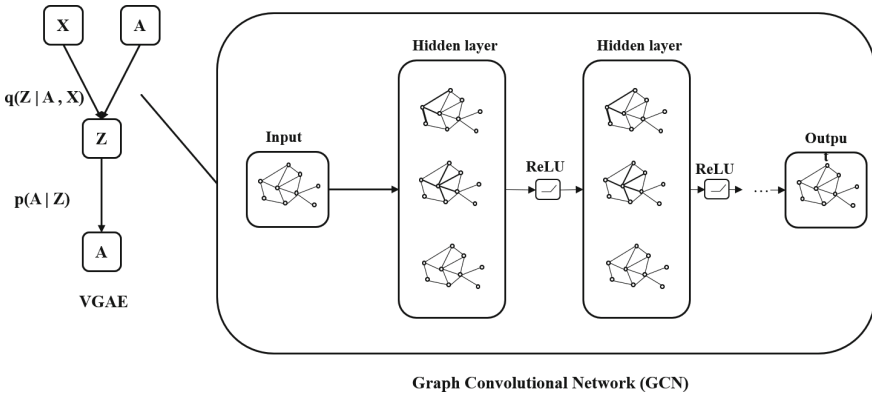


Fig. 4. Graph AutoEncoder (GAE), architecture overview

5.1 Improved AutoEncoder for Graph Embedding

Graph AutoEncoder (GAE) is frequently used in unsupervised learning, which is suitable for learning graph node representations with unsupervised information. To overcome the problem of varied dimensions, we do a complement standardization process for each matrix of dimensions. Meanwhile, we achieve the consistency of the matrix dimensions based on the weight. We classify and compress CFG and blocks for each dimension, and then perform post-normalization matrix processing on the compressed CFG. Lastly, this matrix is trained in the GAE model. Meanwhile, considering that CFG contains structural features and semantic features simultaneously, we innovatively improve the GAE model. In particular, we create two feature matrices independently for the graph embedding, which embeds both structural features and semantic features in the training at the same time. Further experiments reveal that the improved GAE model has high precision and quick convergence. We illustrate the algorithm detail of

the improved Graph AutoEncoder to show how to generate low-dimensional feature vectors. The Graph AutoEncoder realizes the reconstruction of samples by reducing the number of neurons in the hidden layer. In order to duplicate the input data as closely as feasible, the hidden layer of the Graph AutoEncoder must capture the important features of the input data. To put it another way, the Graph AutoEncoder must find the primary components that can represent the original data. Compared with the normal AutoEncoder, the GAE has two different structures. 1) GAE uses an $n \times n$ convolution kernel in the encoder process; 2) GAE has no data decoding part, instead, it has a graph decoder. Meanwhile, GAE adjusts the adjacency matrix and the loss computation. Except for the main differences, GAE can be used for latent vector generation, just like the normal AutoEncoder. Moreover, GAE can be used for link prediction in recommendation tasks.

The following diagram illustrates the kernel concepts for the Improved GAE. Initially, the Variational AutoEncoder (VAE) [24] extracts the feature vectors for the composite graph, which can generate the embedding vectors as well as make the compression for the graph. However, GAE can not directly apply the method of VAE due to the irregular graph-structured data. Each graph has unordered nodes of variable size, and each node in the graph has a different number of neighbors, so we can no longer directly use convolutions to process control flow graphs that contain features of nodes. We use the following steps for the graph data process.

1) In GAE, the encoder is a two-layer graph convolutional network. The first layer is shown in the following equation, in which \bar{A} is the symmetrically normalized adjacency matrix.

$$\bar{X} = GCN(X, A) = ReLU(\tilde{A}XW_0), \tilde{A} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}} \quad (5.1.a)$$

The second Graph Convolution Network (GCN) layer generates μ and $\log\sigma^2$, where

$$\mu = GCN_{\mu}(X, A) = \tilde{A}\bar{X}W_1, \log\sigma^2 = GCN_{\sigma}(X, A) = \tilde{A}\bar{X}W_1 \quad (5.1.b)$$

Now if we combine the math of two-layer GCN together, we get

$$GCN(X, A) = \tilde{A}RELU(\tilde{A}XW_0)W_1 \quad (5.1.c)$$

which generates μ and $\log\sigma^2$. Then we can calculate Z using parameterization trick.

$$Z = \mu + \sigma * \varepsilon, \varepsilon \sim N(0, 1) \quad (5.1.d)$$

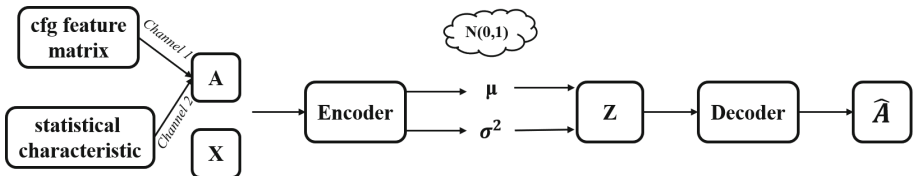


Fig. 5. GAE for binary file vector generation

2) The decoder (generative model) is defined by an inner product between latent variable Z . The output of our decoder is a reconstructed adjacency matrix \hat{A} , which is defined as

$$\hat{A} = \sigma(zz^T) \quad (5.1.e)$$

where $\sigma(\cdot)$ is the logistic sigmoid function.

In summary, the encoder is represented as

$$q(z_i|X, A) = N(z_i | \mu_i, \text{diag}(\sigma_i^2)) \quad (5.1.f)$$

and the decoder is represented as

$$p(A_{oj} = 1|z_i, z_j) = \sigma(z_i^T z_j) \quad (5.1.g)$$

3) The loss function accesses the difference between the created graph and the original graph, which is nearly identical to the VAE loss function. It consists of two parts of the function. The first part is the reconstruction loss between the input node feature matrix and the reconstructed node feature matrix. More specifically, it is the binary cross-entropy between the input high-dimensional control flow graph (A) and the output low-dimensional embedded feature vector (\hat{A}) logits. The second part is the KL-divergence between $q(Z|X, A)$ and $p(Z)$, where $p(Z) = N(0, 1)$. It measures how well our $q(Z|X, A)$ matches $p(Z)$.

$$L = E_{q(Z|X,A)}[\log p(A|Z)] - KL[q(Z|x, A)||p(Z)] \quad (5.1.h)$$

5.2 Matrix Generation Through GAE

As shown in Fig. 1, we employ the GAE model for the representation matrix generation for function and binary files. Several feature extraction method, such as machine learning and deep learning, have been used in feature engineering on binary file similarity detection. Moreover, the strong generalization ability models like CNN and RNN can well learn the feature information of space and time. However, such methods suffer low performance in binary similarity analysis due to the unevenly distributed dataset of normal binary files with large/small samples. Binary similarity detection system needs to be able to detect unknown binary file similarity and unknown malicious binary code detection ability, but the supervised depth model can not accurately identify and classify unknown binary files. Automatic encoder outperforms standard deep unsupervised algorithm, especially in representation learning and confrontation training constraints in potential space. These researchers have made significant contributions in the direction of anomaly detection using the depth unsupervised algorithm, which proves the feasibility of similarity comparison between the unsupervised algorithm in the natural language field and the image field. In contrast to the low-dimensional feature extraction of the depth automatic encoder, the unsupervised depth neural network based on the variational self-encoder uses the multivariate Gaussian probability distribution to reconstruct the input compression feature to the hidden space, and the probability density function is used for error evaluation. The overall similarity detection performance is superior than that of the simple automatic encoder.

Function Matrix Generation: In this stage, we generate the representation matrix for function of the binary file. The function itself contains multiple fundamental blocks, each of which has its own mnemonic and operand. After we use the SkipGram model to generate vector for each block, now we have both the graph adjacent matrix and feature vector for each node in this graph. The adjacent matrix and the feature matrix are used as the input to the GAE model in this case. Through this process, we could find that the compressed data in the GAE model is exactly the fused representation matrix for this function, which contains both semantic information and structure information. Then we perform additional analysis for this matrix to make the binary file similarity detection.

Binary File Matrix Generation: In this stage, we aim to construct the representation matrix for the whole binary file. We make three main types of fuse matrix according the function number in each binary file, taking into account the unevenly distributed data for each binary file (different file has different file size and different number of functions and blocks). In particular, we make the small, medium and large for binary file with function numbers larger than 10, 100 and 1000. The binary file with number less than 10 is also classified as the tiny type. In the binary file, apart from the semantic information represented by the fused matrix, there exists another important information, the statistical feature of function characteristics, which describes another aspect of function feature. As a result, we have two feature vectors for each node in the function call diagram. Instead of vector concatenation directly, we employ the dual channel solution for the Graph AutoEncoder, as shown in Fig. 5. Here we generate the adjacent matrix for the function call diagram, with the function caller as the in-degree and the function called as the out-degree. For each node in this binary file graph, it has specifically two feature vectors, the compressed semantic and structure feature from GAE and the statistical characteristic feature. The second GAE receives the two-channel feature matrix and the adjacent matrix to make a compressed representation matrix for this entire binary file. Next this generation, we make further analysis on the compressed data in the following section.

6 Similarity Comparison

The purpose of code comparison is to find a graph embedding-based matching solution, which maximizes the similarity of the two input binaries. A spontaneous choice is to perform a linear assignment based on the basic block embedding to produce the best match. This strategy, however, has two fatal limitations. 1) Inefficiency: the binary file may contain a huge number of blocks, requiring a long computation time; 2) Accuracy: the linear assignment ignores any graph information and file header information, resulting in a greater error rate. Here we present a new comparison algorithm, the multi-layer perceptron, which can achieve high accuracy in an efficient manner.

6.1 The Design of Multi-layer Perceptron

The multi-layer perceptron and the similarity measurement network consists of two parts: the feature splicing and the fully connected neural network. Specifically, the two

graph feature embedding vectors \vec{g}_1 and \vec{g}_2 are firstly spliced to form a new feature vector. Then the concatenated feature vector gets input into the neural network, which has two full connection layers. The first layer uses ReakyReLU/ReLU activation function, whereas the output layer does not use activation function. Consequently, this network produces a scalar that denotes the relationship between the two graph feature embedding vectors. To put it another way, this scalar represents the similarity between two binary files, which is exactly the similarity score.

As previously stated, the multi-layer perceptron is a network structure composed of two fully connected layers, and the final output is only one scalar. We use the following Equation to represent this process:

$$\mathbf{x} = \mathbf{g}_1 \parallel \mathbf{g}_2 \quad (6.1.a)$$

In this Equation, \mathbf{g}_1 and \mathbf{g}_2 refer to the two feature embedding vectors generated by the graph attention network in Section ‘Matrix Generation Through GAE’, and \parallel means the cascade operation.

Assuming that the input $\mathbf{x} \in \mathbb{R}^{n \times d}$, $n \times d$ represents the dimension, the output of the hidden layer is \mathbf{H} and $\mathbf{H} \in \mathbb{R}^{n \times d}$. Since both the hidden layer and the output layer are fully connected layers, the weight coefficient and bias parameters of the hidden layer can be set as $\mathbf{W}_h \in \mathbb{R}^{d \times h}$ and $\mathbf{b}_h \in \mathbb{R}^{1 \times d}$, the weight and bias parameters of the output layer are $\mathbf{W}_o \in \mathbb{R}^{h \times q}$ and $\mathbf{b}_o \in \mathbb{R}^{1 \times q}$. Then the hidden layer calculation is shown in Eq. (6.1.b).

$$\mathbf{H} = \sigma(\mathbf{x}\mathbf{W}_h + \mathbf{b}_h) \quad (6.1.b)$$

In this Equation, σ indicates the activation function LeakyReLU/ReLU, and the output layer is calculated as shown in Eq. (6.1.c).

$$m = \mathbf{H}\mathbf{W}_o + \mathbf{b}_o \quad (6.1.c)$$

Since the final output layer does not use the activation function and merely conducts simple dimensionality reduction, the similarity scores of the final two embedding vectors are as follows.

$$score(\mathbf{g}_1, \mathbf{g}_2) = m \quad (6.1.d)$$

6.2 The Objective Function

The training objective function of the model is calculated based on the following loss function.

$$L = \frac{1}{k} \sum_{i=1}^k (score_i - y_i)^2 + \lambda \|\mathbf{W}\|_2^2 \quad (6.2.a)$$

In Eq. (6.2.a), k indicates that there are k pairs of binary files, $score_i$ denotes the similarity score of the i pair of binary files, and y_i means the similarity label of the i pair of files. The first part of the function represents the mean square error loss, and the second part is the regularization term to prevent the model from overfitting.

6.3 The Similarity Detection Algorithm of Binary Code

Combining the two improved methods given above, we propose a new similarity detection algorithm for binary code based on graph embedding representation. The detection algorithm is shown below. The entire binary code similarity detection process is divided into four steps:

1. Input n pairs control flow chart $\langle g_1, g_2 \rangle$ of binary functions, perform unsupervised feature extraction through SkipGram model, and extract corresponding function instruction features.
2. Combine the extracted instruction features and flowcharts into the graph embedding generation network, and generate the corresponding graph embedding vectors $\langle \vec{g}_1, \vec{g}_2 \rangle$.
3. Perform feature fusion on each pair of graph embedding vectors as the input of the multi-layer perceptron.
4. Use the multi-layer perceptron neural network to learn the similarity measure, and get the similarity score of each pair of binary functions.

7 Results

This section assesses the performances of CodeDiff on binary files, which consist of two parts, the cross-version files and cross-optimization-version files. We investigate the dataset of OpenSSL [25] and Compiler [26] to demonstrate the usefulness and effectiveness of CodeDiff. Moreover, we analyze the cross-architecture binary diff performance on the case study, utilizing the FFMpeg dataset.

Table 1. Diff results for cross version binary files

Cross version		Precision				Recall			
		BinDiff	Gemini	VulSeeker	CodeDiff	BinDiff	Gemini	VulSeeker	CodeDiff
OpenSSL	fips2.0.16-1.1.1m	0.769	0.640	0.601	0.785	0.521	0.665	0.638	0.717
	0.9.6l-1.1.1m	0.775	0.661	0.638	0.832	0.582	0.639	0.681	0.764
	1.0.0s-1.1.1m	0.790	0.739	0.780	0.928	0.778	0.795	0.776	0.908
	1.1.0l-1.1.1m	0.856	0.750	0.782	0.922	0.767	0.809	0.803	0.912
	Average	0.798	0.698	0.700	0.867	0.662	0.727	0.725	0.825
Curl	7.42.0-7.82.0	0.672	0.721	0.746	0.822	0.601	0.730	0.724	0.802
	7.52.0-7.82.0	0.771	0.843	0.919	0.936	0.795	0.839	0.806	0.930
	7.62.0-7.82.0	0.934	0.912	0.923	0.953	0.898	0.891	0.917	0.941
	7.72.0-7.82.0	0.946	0.931	0.914	0.966	0.934	0.927	0.897	0.938
	Average	0.831	0.849	0.876	0.919	0.807	0.847	0.836	0.903

Experimental Setup: Without sacrificing generality, we make our experiments on a modern operating system, DeepIn 20.3, which is a debian based Linux system. Our computer has a CPU of I7-10700K with an NVIDIA gpu of RTX2080, as well as 32 GB memory. Furthermore, we make extra experiments on the Raspberry Pi 4 with 4 GB memory to verify the binary diff, which also shows the lightweight of our method.

Baseline Solution and Groundtruth: Here we compare CodeDiff to numerous state-of-art binary diff technologies, such as BinDiff, Gemini and VulSeeker [27]. To get the groundtruth for binary file difference, we employ the source code analysis to gain the code line similarity detection.

Metrics: generally, we utilize the following metrics to evaluate the performance of CodeDiff, the precision rate, the recall rate and the unit process time for binary file. Precision is the fraction of relevant instances among the retrieved instances, while recall is the fraction of relevant instances that were retrieved. The unit process time for binary file means the real process time to represent a binary file to an embedded vector, which could be used directly in the multi-layer perception to get the similarity score.

7.1 Diffing Performance

We implement the prototype of CodeDiff and analyze the diff performance through the metrics of precision and recall. We make the comparison for FFMpeg dataset from three angles, the cross-version and cross-optimization analysis for OpenSSL and Curl datasets, and the cross-architecture analysis. We download the latest version of those datasets (in the time of this paper was written) and compile them using the gcc-v9.4 compiler.

Cross-Version Diffing. First, we take advantage of four separate tools to differentiate the cross version binary files with default optimization (O1 level) for datasets OpenSSL and Curl, and the results are displayed in Table 1. It is not difficult to find that our method, CodeDiff, achieves the average highest precision and recall among all diff tools. E.g., for the OpenSSL diff between version 0.9.6l-1.1.1m, CodeDiff achieves the 0.832 precision with 0.764 recall, which improves at least 10% than other three methods. Meanwhile, when the two different binary files have slight differences, CodeDiff achieves both high precision and recall, implying that CodeDiff can make successful analysis for binary files. We can also see that the analysis tool Gemini and VulSeeker have higher recall than BinDiff, which proves that additional semantic information helps improve the quality of binary file diff.

Cross-Optimization Diffing. In this section, we run tests to evaluate the CodeDiff performance on different optimization methods, using the datasets of OpenSSL and Curl. We compile those datasets on our Deepin OS with multiple optimization levels, such as O0, O1, O2 and O3. The results are then recorded and reported in Table 2. We can plainly find that CodeDiff achieves the maximum precision and recall performance in most cross optimization cases. However, in the case of O2-O3 in OpenSSL 3.0.1 and O2-O3 in Curl 7.30.0, BinDiff achieves the highest Precision and Recall, separately. While CodeDiff achieves a smaller precision and recall, which comes from the small number of binary files. It is hard to extract accurate semantic information from only a few binary files and embed them into the adjacent matrix. However, even in such special cases, CodeDiff also achieves high diff performance in both precision and recall, which proves that CodeDiff can successfully make the comparison for distinct optimization binary files.

Table 2. Diff results for cross optimization binary files

Cross optimization			Precision				Recall			
			BinDiff	Gemini	VulSeeker	CodeDiff	BinDiff	Gemini	VulSeeker	CodeDiff
OpenSSL	3.0.1	O0-O3	0.291	0.279	0.272	0.324	0.133	0.15	0.15	0.299
		O1-O3	0.631	0.506	0.467	0.674	0.519	0.538	0.528	0.626
		O2-O3	0.947	0.819	0.871	0.914	0.82	0.863	0.868	0.933
	1.1.1m	O0-O3	0.327	0.295	0.291	0.405	0.16	0.23	0.209	0.401
		O1-O3	0.698	0.606	0.621	0.775	0.504	0.604	0.634	0.689
		O2-O3	0.932	0.855	0.834	0.951	0.857	0.919	0.903	0.935
Curl	7.82.0	O0-O3	0.138	0.14	0.149	0.249	0.082	0.141	0.136	0.302
		O1-O3	0.748	0.714	0.671	0.909	0.685	0.695	0.718	0.842
		O2-O3	0.92	0.936	0.942	0.967	0.862	0.929	0.878	0.975
	7.30.0	O0-O3	0.136	0.171	0.184	0.325	0.074	0.149	0.142	0.302
		O1-O3	0.649	0.723	0.698	0.841	0.553	0.639	0.618	0.79
		O2-O3	0.91	0.932	0.908	0.948	0.964	0.93	0.949	0.947

Table 3. Diff results for cross architecture binary files

Cross architecture		Precision				Recall			
		BinDiff	Gemini	VulSeeker	CodeDiff	BinDiff	Gemini	VulSeeker	CodeDiff
ffmpeg (7:4.3.3-0+deb11u1)-amd64	arm64	0.742	0.908	0.757	0.913	0.552	0.802	0.536	0.882
	armhf	0.793	0.803	0.8	0.838	0.777	0.794	0.704	0.781
	i386	0.885	0.939	0.814	0.926	0.857	0.887	0.768	0.952
	mipsel	0.679	0.827	0.643	0.862	0.587	0.763	0.531	0.846

Cross-Architecture Diffing. More experiments are conducted here to verify that the semantic-based method, CodeDiff, can perform diff on even different CPU architectures, e.g., AMD64, ARM64, x86. We download the latest FFMpeg from the debian repository with different CPU architectures to perform the cross architecture diffing, as indicated in Table 3. In the cross architecture diffing, we can find that CodeDiff achieves the best performance on both precision and recall, which comes from the accurate semantic information extraction and function graph building. We can also see that the amd64 architecture has the highest similarity with the i386 architecture among all diff tools, which is in line with the intuition that 32bit program and 64bit program has slight differences.

7.2 Model Parameter Selection

In CodeDiff, there are two important structures, the Word2vec model, as known as the SkipGram method, and the Graph AutoEncoder (GAE). The building of corpus highly relies on the SkipGram, which depends more on the word itself instead of the hyper parameters in SkipGram. Here we conduct experiments about changing hyper parameters in the GAE to investigate the relationship between the binary diff performance and hyper parameters in GAE. Specifically, we change the factors of dropout rate and a number of units in the hidden layer, to demonstrate the diff performance in the cross-version dataset OpenSSL and Curl with default gcc optimization (O1).

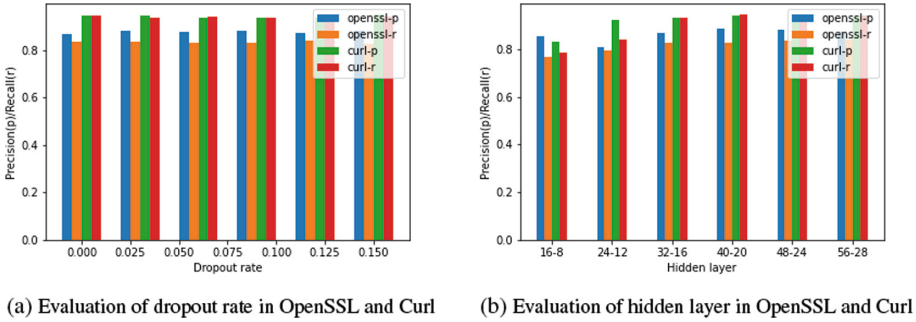


Fig. 6. Hyper parameters selection in CodeDiff

Dropout Rate. As shown in Fig. 6a, we analyze the relationship between dropout in GAE and model predict accuracy (precision and recall). We can immediately find that there have slightly performance changes along with the dropout rate, indicating that our GAE model does learn the characteristic of binary files and there is no obvious overfitting detected. Moreover, to assurance CodeDiff has the highest adaptability for different datasets, here we set the dropout rate to zero.

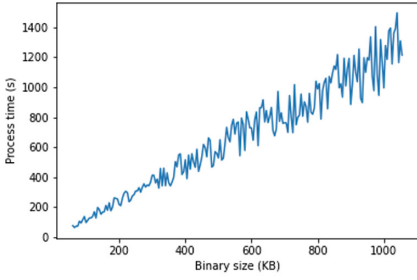
Hidden Layer. In addition, we change the number of hidden layers in the GAE model to decide which type of hidden layer has the best performance in the binary diff task, the result is shown in Fig. 6b. Along with the enlargement of the number of the hidden layer, we can find the precision and recall both growth. However, when the hidden layer is larger than ‘32-16’, the precision and recall of datasets benefit little, but the computation time for the GAE grows quickly, which reduces the effectiveness of CodeDiff. Therefore, we select the ‘32-16’ as our hidden layer to attain high accuracy while also computing efficiency.

7.3 Effectiveness

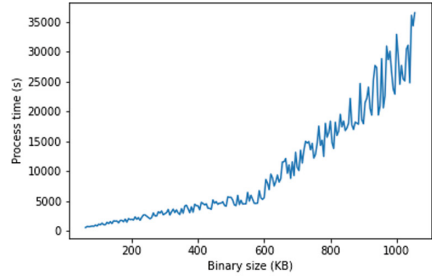
This section makes extra study for CodeDiff in the data process to prove the effectiveness of our method. Specifically, we analyze the processing time for each step in CodeDiff. Moreover, we transplant CodeDiff to Raspberry Pi to process data, which further demonstrates the lightweight of CodeDiff.

Process Time. This section details the process time of the binary file and the training time of the model.

1. **Model Training:** we have a total of four models to be trained, SkipGram, GAE for basic blocks, GAE for finary files and Multi-layer perceptron. Since this work is a one-time effort, we are more concerned with the prediction accuracy rather than the training time, which results in 100–300 epoches for loss converges for each model. This training time totally consumes 11 h in our DeepIn OS, and it can be accelerated by using more advanced CPU and GPU.



(a) Binary matrix generation time on Deepin



(b) Binary matrix generation time on Raspberry Pi

Fig. 7. Binary matrix generation time on different platform

2. **Data Preparation:** this step extracts the instruction and operand information from raw binary file using the IDA pro tool. Each binary file takes average of 6.9 s to extract the information required by CodeDiff.
3. **Block Matrix Generation:** in this stage, we transform the word (instruction and operand) in blocks to vector referring to the pre-build word corpus by the SkipGram model. Then we embed this vector to the adjacent matrix inside the function to fuse the CFG Matrix. When processing a binary file with 194 identical functions, we make use of 1.3 s to generate the matrix for each block.
4. **Binary Matrix Generation:** we use the improved GAE model to embed the CFG feature matrix and statistic feature to the function call graph and generate the binary matrix. This processing time highly depends on the size of the binary file. We report the detailed relation between generation time and file size in Fig. 7a. We can clearly see that there is a linear relationship that represents the process time of CodeDiff is almost a constant time for each unit size of binary file. Specifically, we use an average of 1.2 s to process 1kb size of the binary file.
5. **Similarity Comparison:** here we employ the multi-layer perceptron to measure the similarity for each pair of binary functions, and then we generate the precision and recall for the binary file comparison. This process uses an average of 20 s for a pair of binary files with 50 kb size. From the process time evaluation of the above five steps, CodeDiff exhibits high efficiency for binary file diffing. We further proves the lightweight of our method in the following sections.

Evaluation on Raspberry Pi. In this section, we port our model from Desktop Computer to an Edge Computing Node, the Raspberry Pi, to further demonstrate the great efficiency and lightweight of our model. We evaluate the unit processing time for block matrix generation, binary file matrix generation and similarity comparison, as shown in Fig. 7b. Specifically, our Raspberry Pi runs the Pi OS version 5.10.63-v8+ in aarch64 architectures, which has 4 GB ROM and 32 GB RAM. Moreover, we increase the SWAP size to 4 GB to prevent the run out of memory. CodeDiff employs an average of 4.9 s to process the block matrix. For binary matrix, CodeDiff utilizes an average of 9.7 s to process 1kb size of binary file. However, this time increases to 29.6 s when the size binary file gets larger than 600kb, which comes from the memory lack of ROM and

the SWAP memory does decrease the processing time. A large ROM of a faster disk for SWAP memory will help improve this problem. And CodeDiff uses an average of 97 s for similarity comparison to a pair of binary files with 50 kb size. This evaluation on Raspberry Pi proves that CodeDiff achieves both high efficiency and lightweight in binary file similarity detection.

8 Related Work

Program Logic-Based Detection. Based on the program logic, this detection technology uses data structures, such as the lists, trees and graphics, to record and describe the data flow and control flow information of the program. Simply put, the intermediate representation can capture certain syntax and semantic information in the program, this technology matches the similar sequences and subgraphs to find logically and functionally similar program basic blocks. This program logic is reflected as the data flow inside the function. It is also reflected as the input/output of the function. For example, Multi-MH system [1] used the input/output of the function to grasp the semantics, and it took advantage of the signatures to find vulnerable code with similar behavior. The binary search engine, Bingo, utilized input/output samples generated from symbolic expressions to match semantically similar functions.

The program control logic can be described by function call sequence outside function level, control flow graph (CFG) inside function level and logic tree in basic block level. For example, the tool HAWK [2] implemented a dynamic detection method based on the System Call Dependence Graph (SCDG) birthmark. The TEDEM method proposed by Pewny [3] used the edit distance of the expression tree to measure the similarity in the basic block level. The well-known binary similarity detection systems, discovRE [4] and Genius [5], applied the optimized graph matching algorithm based on attribute-optimized CFG. Meanwhile, tools like BinHun [6] and iBin-Hunt [7] were also based on CFG, and they utilized the symbolic execution to determine the similarity of basic blocks and functions. CFG is a high degree of abstraction of the program code, which means it could effectively capture and display the control flow information of the program. In addition, CFG is an intermediate representation, which can be used to represent both source code and assembly code. Due to the cross-language nature of CFG, most similarity detection technology rely on CFG heavily, especially for cross-architecture binary code detection. The program logic-based detection have the advantages of high accuracy, strong scalability and flexible matching algorithms and strategies. However, there exists fatal limitation, which is the high computing requirement. There is an expensive process for the extraction of data flow and CFG. Besides, the graph matching algorithm lacks for a polynomial solution, which means that the amount of calculation increases exponentially with the size of the code base.

Semantic-Based Detection. Semantic-based detection technology compares the semantic differences between functions and components. By capturing the semantic information in the program assembly code, it achieves similarity measurement. Such a method generally draws on NLP, image recognition and other technologies. It uses

deep neural networks (DNN) to embed program semantics, and then implements large-scale task processing by comparing or querying embedded vectors. The main intermediate representation for binary code includes three parts: normalized assembly text, other intermediate languages and CFG. The neural network model mostly adopts the siamese architecture, which uses the large-scale feature of the program code to train the sample library. This implies that the experience knowledge is not necessary for the DNN. Xu et al. [8] proposed the first method of CFG embedding for binary functions based on neural networks. In the Gemini system, they used the improved Structure2ve model to construct the Siamese architecture network, which embedded the CFG in a high-dimensional vector and calculated the similarity between functions by the cosine distance of the vector. The Structure2ve model was derived from the graphical model inference algorithm, which aggregated the vertex-specific feature vector from the topological structure of the graph in a recursive manner. Primarily, the PV-DM model was designed for text data, and it learnt document representation based on the identification of the document. Then Ding et al. put forward the Asm2vec [9] model, which was an assembly code representation learning model based on an improved PV-DM model [16]. The Asm2vec model made use of the mechanism of custom function inlining and random walk to model the CFG as a linear sequence of assembly instruction. It took the assembly text as input, learnt the instruction semantics, built the instruction embedding vector, and got the semantic embedding vector of the function without any prior knowledge. As the first scheme which used representation learning as assembly code to construct feature vectors, the Asm2vec provided outstanding anti-obfuscation and anti-compiler optimization properties. However, it was not suitable for cross-architecture comparison. The Word2vec [13] was a Google NLP tool with the ability to vector words and quantitatively measured the relationship between words. The SAFE network [10] then took advantage of the Word2vec model of NLP to implement instruction embedding in assembly language firstly. Then it utilized recurrent neural network (RNN) [17] to capture the context of the instruction sequence and embed the functions. The SAFE abandoned CFG as the intermediate representation. Instead, the semantic information of assembly code was directly embedded into high-dimensional vectors by DNN, which not only saved the time-consuming CFG extraction process, but also avoided the introduction of human bias. Based on the RNN architecture, the SAFE model used the supervised learning method to train the model, which could automatically construct positive sample pairs and sub-sample pairs randomly. However, for the cross-architecture detection tasks, with the increase of system instruction architectures, the size of the training sample library needed to be expanded exponentially. This was a fatal limitation for the SAFE model in the scalability. As the first method to use a semantic representation to learn a binary code intermediate language, GeneDiff [11] enabled cross-architecture clone detection. It relied on dynamic analysis of VEXIR's intermediate language [18], which eliminated the differences between different instruction architectures. It generated semantics for the VEXIR to embed function vectors through an improved PV-DM model. Since each assembly instruction will be translated into multiple VEX instructions, the model regarded the combination of multiple VEX instructions translated from the same assembly instruction as a word. The basic bloc was also treated as a sentence,

whereas the function was regarded as a paragraph. Finally, GeneDiff used the cosine distance between vectors to measure the similarity between functions.

In addition to the improvement of detection speed and accuracy, the advantage of applying neural networks to similarity detection tasks is flexibility. The matching algorithms used in traditional detection methods are usually fixed, but the neural network can be retrained for different tasks. Moreover, the neural network can not only learn and select the features automatically, but it can also learn the weights of distinct features, which is difficult to determine by artificial methods, thereby reducing or even avoiding the overfitting caused by manual design and features extraction.

Defects on Existing Works. Despite these positives, we identify three key drawbacks of existing learning-based methods. First, current learning-based methods are unable to achieve efficient program-wide binary difference on block-level in a fine-grained basis. Most methods analyze the difference on the function-level, without delving into the similarity inside the function structure. Additionally, those methods rely heavily on fine-grained frameworks and efficient difference tools. Only in this way, can those methods conduct a comprehensive and effective similarity comparison. Second, representations are not comprehensive enough for feature selection. For example, InnerEye [12] extracted basic block semantic information using NLP techniques and the longest common subsequence, but it only considered dependency information in local and small control code components. Asm2Vec only generated random walks inside functions to learn token and function embeddings. Third, there is a lack of a general algorithm which can be both suitable across architecture and common binary files. Because of the extreme diversity of binary files, supervised learning may be affected by overfitting. Meanwhile, considering the entire instruction as an object may result in serious out-of-vocabulary problems (OOV) [19].

9 Conclusions

In this paper, we propose CodeDiff, a new technique for malware vulnerability detection on IoT and edge computing platforms that employs both semantic and structure information to detect binary similarity. Besides, CodeDiff is an unsupervised learning method which does not require label data. Through the SkipGram with Negative Sampling, we build the word vocabulary for instruction data. Then we use the Graph AutoEncoder to embed both the semantic and structure information to the representation matrix for the CFG. After all, we utilize the improved Graph AutoEncoder to fuse all the function structure, function characteristics and function features to the fusion matrix. Finally, we present the specific matrix comparison to achieve a high accuracy similarity result in fast speed. Furthermore, we make an experiment for the prototype on binary datasets OpenSSL and Curl with a case study of FFMpeg, which shows that CodeDiff gives a high performance on the binary file similarity detection.

Acknowledgments. This work was supported in part by the National Natural Science Foundation of China (Grant No. 62072453, 61972392) and the Youth Innovation Promotion Association of the Chinese Academy of Sciences (Grant No. 2020164).

References

1. Pewny, J., Garmany, B., Gawlik, R., Rossow, C., Holz, T.: Cross-architecture bug search in binary executables. In: IEEE Symposium on Security and Privacy 2015, pp. 709–724 (2015)
2. Wang, X., Jhi, Y.-C., Zhu, S., Liu, P.: Behavior based software theft detection. In: Proceedings of the 16th ACM Conference on Computer and Communications Security, pp. 280–290 (2009)
3. Tian, J., Xing, W., Li, Z.: BVDetector: a program slice-based binary code vulnerability intelligent detection system. *Inf. Softw. Technol.* **123**, 106289 (2020)
4. Eschweiler, S., Yakdan, K., Gerhards-Padilla, E.: discoverRE: efficient cross-architecture identification of bugs in binary code. In: NDSS, vol. 52, pp. 58–79 (2016)
5. Feng, Q., Zhou, R., Xu, C., Cheng, Y., Testa, B., Yin, H.: Scalable graph-based bug search for firmware images. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 480–491 (2016)
6. Gao, D., Reiter, M.K., Song, D.: BinHunt: automatically finding semantic differences in binary programs. In: Chen, L., Ryan, M.D., Wang, G. (eds.) ICICS 2008. LNCS, vol. 5308, pp. 238–255. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-88625-9_16
7. Ming, J., Pan, M., Gao, D.: iBinHunt: binary hunting with inter-procedural control flow. In: Kwon, T., Lee, M.-K., Kwon, D. (eds.) ICISC 2012. LNCS, vol. 7839, pp. 92–109. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37682-5_8
8. Xu, W., Evans, D., Qi, Y.: Feature squeezing: Detecting adversarial examples in deep neural networks. arXiv preprint [arXiv:1704.01155](https://arxiv.org/abs/1704.01155) (2017)
9. Ding, S.H., Fung, B.C., Charland, P.: Asm2Vec: boosting static representation robustness for binary clone search against code obfuscation and compiler optimization. In: IEEE Symposium on Security and Privacy (SP), pp. 472–489 (2019)
10. Massarelli, L., Di Luna, G.A., Petroni, F., Baldoni, R., Querzoni, L.: SAFE: self-attentive function embeddings for binary similarity. In: Perdisci, R., Maurice, C., Giacinto, G., Almgren, M. (eds.) DIMVA 2019. LNCS, vol. 11543, pp. 309–329. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-22038-9_15
11. Luo, Z., Wang, B., Tang, Y., Xie, W.: Semantic-based representation binary clone detection for cross-architectures in the internet of things. *Appl. Sci.* **9**(16), 3283 (2019)
12. Zuo, F., Li, X., Young, P., Luo, L., Zeng, Q., Zhang, Z.: Neural machine translation inspired binary code similarity comparison beyond function pairs. arXiv preprint [arXiv:1808.04706](https://arxiv.org/abs/1808.04706) (2018)
13. Church, K.W.: Word2Vec. *Nat. Lang. Eng.* **23**(1), 155–162 (2017)
14. Eagle, C.: The IDA Pro Book. No Starch Press (2011)
15. Andriessse, D.: Practical Binary Analysis: Build Your Own Linux Tools for Binary Instrumentation, Analysis, and Disassembly. No Starch Press (2018)
16. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: International Conference on Machine Learning, pp. 1188–1196 (2014)
17. Zaremba, W., Sutskever, I., Vinyals, O.: Recurrent neural network regularization. arXiv preprint [arXiv:1409.2329](https://arxiv.org/abs/1409.2329) (2014)
18. Zhao, J., Nagarakatte, S., Martin, M.M., Zdancewic, S.: Formalizing the LLVM intermediate representation for verified program transformations. In: Proceedings of the 39th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, pp. 427–440 (2012)
19. Hetherington, I.L.: A characterization of the problem of new, out-of-vocabulary words in continuous-speech recognition and understanding (1995)
20. L. DigitalOcean: ODA - the online disassembler, November 2021. <https://www.online-disassembler.com>

21. Lai, S., Liu, K., He, S., Zhao, J.: How to generate a good word embedding. *IEEE Intell. Syst.* **31**(6), 5–14 (2016)
22. Blajer, J.A.G.W., Krawczyk, M.: The inverse simulation study of aircraft flight path reconstruction. *Transport* **17**(3), 103–107 (2002)
23. Pan, S., Hu, R., Long, G., Jiang, J., Yao, L., Zhang, C.: Adversarially regularized graph autoencoder for graph embedding. arXiv preprint [arXiv:1802.04407](https://arxiv.org/abs/1802.04407) (2018)
24. Kusner, M.J., Paige, B., Hernández-Lobato, J.M.: Grammar variational autoencoder. In: *International Conference on Machine Learning*, pp. 1945–1954 (2017)
25. T. O. P. Authors: OpenSSL, November 2019. <https://www.openssl.org/>
26. Cooper, K.D., Torczon, L.: *Engineering a Compiler*. Elsevier, New York (2011)
27. Gao, J., Yang, X., Fu, Y., Jiang, Y., Sun, J.: VulSeeker: a semantic learning based vulnerability seeker for cross-platform binary. In: *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 896–899 (2018)
28. Nagra, J., Collberg, C.: *Surreptitious Software: Obfuscation, Watermarking, and Tamperproofing for Software Protection: Obfuscation, Watermarking, and Tamperproofing for Software Protection*. Pearson Education (2009)
29. Cui, A., Costello, M., Stolfo, S.: *When firmware modifications attack: a case study of embedded exploitation* (2013)
30. Martin, A., Raponi, S., Combe, T., Di Pietro, R.: Docker ecosystem-vulnerability analysis. *Comput. Commun.* **122**, 30–43 (2018)



Multi-dimensional Data Quick Query for Blockchain-Based Federated Learning

Jiayi Yang^{1,2}, Sheng Cao^{1,2}(✉), Peng Xiangli³, Xiong Li^{1,2},
and Xiaosong Zhang^{1,2}

¹ Shenzhen Institute for Advanced Study, University of Electronic Science
and Technology of China, Shenzhen, China

caosheng@uestc.edu.cn

² School of Electrical and Computer Engineering, University of Electronic Science
and Technology of China, Chengdu, China

³ The Fifth Electronic Research Institute of MIIT, Guangzhou, China

Abstract. Due to the drawbacks of Federated Learning (FL) such as vulnerability of a single central server, centralized federated learning is shifting to decentralized federated learning, a paradigm which takes the advantages of blockchain. A key enabler for adoption of blockchain-based federated learning is how to select suitable participants to train models collaboratively. Selecting participants by storing and querying the metadata of data owners on blockchain could ensure the reliability of selected data owners, which is helpful to obtain high-quality models in FL. However, querying multi-dimensional metadata on blockchain needs to traverse every transaction in each block, making the query time-consuming. An efficient query method for multi-dimensional metadata in the blockchain for selecting participants in FL is absent and challenging. In this paper, we propose a novel data structure to improve the query efficiency within each block named MerkleRB-Tree. In detail, we leverage Minimal Bounding Rectangle (MBR) and bloom-filters for the query process of multi-dimensional continuous-valued attributes and discrete-valued attributes respectively. Furthermore, we migrate the idea of the skip list along with an MBR and a bloom filter at the head of each block to enhance the query efficiency for inter-blocks. The performance analysis and extensive evaluation results on the benchmark dataset demonstrate the superiority of our method in blockchain-based FL.

Keywords: Federated learning · Blockchain · Multi-dimensional query

1 Introduction

As a special distributed machine learning framework, FL, in which allows multiple data owners to train machine learning models collaboratively with their data stored locally, is much popular in the present age [1]. However, centralized FL still faces some challenges such as the failure of a single central server, etc.

With the overwhelming development of blockchain technology, it is possible to leverage some advantages of blockchain to FL and construct a decentralized FL paradigm named blockchain-based FL [2,3]. In blockchain-based FL, blockchain is able to enhance the robustness, trust, security of FL, as well as providing a credible cooperation mechanism among participants.

When the aggregation server initializes a FL task, it need to select a set of data owners to participate. Selecting participating nodes according to their data type without knowing the metadata information of data owners is challengeable. Through providing secure data storage platform in blockchain-based FL, data owners can announce the description of their data called metadata in the community via blockchain [4]. When the metadata is queried on the blockchain, the aggregation server can obtain the candidate participants list from nearest proxy server and invite these nodes to participate in FL [5]. And we will introduce the process in Sect. 3.

However, on the one hand, the query efficiency on the existing blockchain is extremely low [6]. With the increasing number of data owners registering, it cannot meet the large query requirements of the aggregation server when selecting nodes. On the other hand, in the real scenario of choosing data owners in FL, the query condition may usually be composed by multi-dimensional continuous-valued attributes and discrete-valued attributes [4]. Existing query methods on the blockchain can only cater for single-dimensional hash value. It is inefficient to store multi-dimensional attributes in the single dimensional data structures, since it needs to do intersectional operation when we need to query for multi-dimensional attributes [7]. For multi-dimensional query condition with both continuous-valued attributes and discrete-valued attributes on the blockchain, yet there is no appropriate query method to satisfy this kind of query demand [8]. In this paper, we propose a method for the query process of both inter-block and intra-block. Our contributions are listed in the following points:

- We formulate the selection of participating nodes in blockchain-based FL as the metadata query problem on blockchain. We divide this query problem into intra-block query and inter-block query and put forward schemes for them respectively.
- For intra-block query, we modify the structure of the block and construct an MerkleRB-Tree in each block. Query schemes for both discrete-valued attributes and continuous-valued attributes are proposed.
- For inter-block query, we apply the skip list and implement an inter-block query scheme with bloom-filters and MBR for discrete-valued attributes and continuous-valued attributes respectively.
- We analysis the performance of the query schemes we propose. The results of the comparative experiments with the baseline method show our schemes are more efficient.

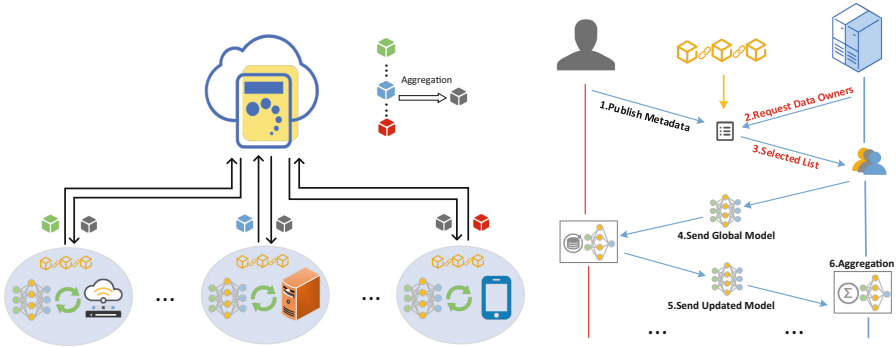


Fig. 1. Blockchain-based FL framework

2 Related Work

2.1 Blockchain Empowered Federated Learning

Since traditional centralized FL faces a number of challenges [9] such as lack of a secure and credible cooperation mechanism etc., an increasing number of studies focus on empowering FL with blockchain [10]. Being empowered with blockchain, FL owns a credible incentive and contribution measurement mechanism as well as strengthens its security [11].

Besides, blockchain provides a trusted storage mechanism for FL, allowing data to be shared securely. Data owners can leverage blockchain to publish their metadata information and then aggregation servers can select participating nodes by querying the metadata on the blockchain according to the data type [5]. Zhang et al. propose a FL protocol based on blockchain in which the nearest proxy server can help to query metadata on blockchain and return the set of selected participating nodes [4]. However, these studies do not focus on the query efficiency of metadata in blockchain-based FL, nor did they change the original block structure.

2.2 Query on the Blockchain

For query on the blockchain, traversing every transaction of each block can be regarded as time-consuming. Current studies show that using external databases can improve the query efficiency of blockchain query [12, 13]. By establishing an efficient query layer, EtherQL, Li et al. propose a quick query method that imports block data into an off-chain database using the Ethereum listening interface [14]. Peng et al. propose a three-tier blockchain query architecture, which saves the time to traverse unnecessary blocks [6]. Zhang et al. design new data structures named *Gem²Tree* which can be effectively maintained by blockchain, significantly reducing the storage and computing cost of smart contract [8]. However, these schemes are hardcoded and cannot be well adapted to different query conditions and do not consider the problem of the inter-block query.

3 Problem Formulation

3.1 System Framework

In the training process of blockchain-based FL, the data owners register the FL community and publish the metadata to the blockchain. It is noticeable that the metadata generally refer to the description of the data type of the data owners. When the metadata is queried by the aggregation server, it can get the candidates list according to the task requirements. Then the aggregation server initializes the machine learning model and allocates it to participants for local training. Finally, after getting the updated models from participants, the aggregation server aggregates them to update the global model. The system paradigm is detailed in Fig. 1.

3.2 Query Metadata on the Blockchain

In our system framework, the aggregation are responsible for querying the metadata on the blockchain. Moreover, when the aggregation servers select parties to participate in the FL task, they usually select different parties to join in based their type of data sets according to the requirements of machine learning model training task. In the metadata, there are discrete-valued attributes and also continuous-valued attributes. Each query condition may contain multi-dimensional discrete-valued attributes and continuous-valued attributes. Therefore, we can regard the problem as the mixed multi-dimensional query of continuous-valued attributes and discrete-valued attributes.

However, it is inefficient to use existing methods to solve this problem. In the traditional way, in order to find the data owners who have this type of data set, they may need to traverse all the transactions in every block and check whether each query condition is satisfied. It is time-consuming if we traverse all the transactions in the blockchain. If the query condition of continuous-valued attributes is multi-dimensional, it will increase the difficulty and cost of querying to a greater extent [7]. Therefore, the problem is how to design a data structure in the blockchain to query both discrete-valued attributes and multi-dimensional continuous-valued attributes more efficiently. In the next two sections, we divide this problem into the intra-block query and inter-block query and describe the solution we propose for this problem respectively.

4 Intra-block Query Scheme

4.1 The Structure of MerkleRB-Tree

In order to make the intra-block query quicker, we apply the idea of MR-Tree [15] and bloom filter together to construct a new structure named MerkleRB-Tree. MerkleRB-Tree extends the advantages of MR-Tree and bloom filter. We use it for multi-dimensional query of both continuous-valued and discrete-valued attributes. As shown in Fig. 2, MerkleRB-Tree verifies the integrity of the whole

tree based on Merkle Hash Tree. Each internal node contains a hash value, a bloom filter, and an MBR [16]. The bloom filter in each node can be used to check whether there are existing transactions in the current subtree that satisfy the discrete-value query condition. MBR covers the range of continuous-valued attributes of all transactions of every dimensions in its subtree.

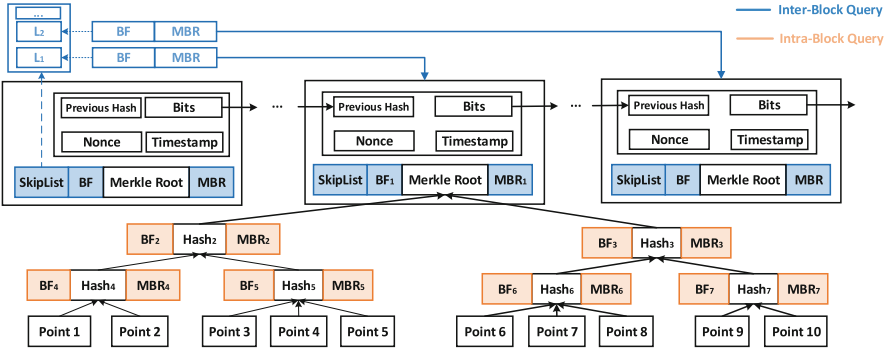


Fig. 2. The query structure for both discrete-valued and continuous-valued attributes on blockchain

4.2 Intra-block Query of Continuous-Valued Attributes

For querying multi-dimensional continuous-valued attributes, it is inefficient to take the intersection after querying the multiple dimensions separately for querying multi-dimensional continuous-valued attributes. In this section, we focus on the query of multi-dimensional continuous-value. In the MerkleRB-Tree, the spatial range of the MBR at the root node of the whole tree. In the query process, we use the recursion method to traverse all the child nodes of the current node. If there is an intersection between the spatial scope of the multi-dimensional query condition and the child node, then we continue to search down the current subtree. The specific algorithm is shown in Algorithm 1. Using the above method for multi-dimensional range query, we can save the time cost of traversing unnecessary nodes in MerkleRB-Tree and improve the efficiency of the query process.

4.3 Intra-block Query of Discrete-Valued Attributes

For querying discrete-valued attributes, we add a bloom filter [17] in each node of MerkleRB-Tree. A Bloom filter is a long binary vector and a series of random mapping functions that can be used to check whether an element is not in a set. In MerkleRB-Tree, the bloom filter of each node can determine whether all the transactions in the subtree do not satisfy the query condition of discrete-valued

attributes. In other words, the non-leaf node's bloom filter is the sum of all its child nodes' bloom-filters, which we represent in formula (1).

$$BF_{parent} = BF_{child}^1 + BF_{child}^2 + \dots BF_{child}^n \quad (1)$$

For each discrete-valued query condition, we start it from the root node of MerkleRB-tree. For all the transactions in the left and right subtrees that do not satisfy the discrete-valued query condition, bloom-filters in each node are used to find these subtrees and not query them.

5 Inter-block Query Scheme

5.1 Inter-block Index Structure

In Fig. 2, an MBR and a bloom filter are added to the block header. For querying discrete-valued attributes, we use the bloom filter at the head of the block to verify whether the block contains any transactions that satisfies the query condition. If no transaction satisfies the query condition, we do not need to query within the block. Similarly, for the inter-block query of the multi-dimensional continuous-valued attributes, we also need to check whether the range space of query condition has an intersection with the MBR at the head of each block.

However, traversing all the blocks in the blockchain is time-consuming. In order to solve this problem, inspired by the idea of dichotomy, we apply a skip list and put forward an efficient inter-block query method. The architecture of the inter-block query is shown in Fig. 2, and each level of the skip list includes a bloom filter and an MBR denoted as $SkipList_{BF}^i$ and $SkipList_{MBR}^i$. For $SkipList_{BF}^i$ which is used to query discrete-valued attributes, the i^{th} level's bloom filter in the skip list can be used to check whether there are satisfied transactions in the next α^i blocks. We can use formula (2) to represent it.

$$BF_{SkipList}^i = BF_{block}^{current} + \dots + BF_{block}^{current+\alpha^i} \quad (2)$$

Algorithm 1 IntraQuery(Node, MBR_{query})

Input: Query condition $q = \langle Q_{discrete}, MBR_{query} \rangle$, query tree *Tree*

Output: result Ω

- 1: **if** Node.isLeaf()=False **then**
 - 2: **for** MBR_{child}^i for Node **do**
 - 3: **if** MBR_{child}^i intersects MBR_{query} AND
 BF_{child}^i .isContain($Q_{discrete}$) **then**
 - 4: IntraQuery(Node.Child(i), MBR_{query})
 - 5: **end if**
 - 6: **end for**
 - 7: **else**
 - 8: Add this to the result set Ω
 - 9: **end if**
-

Similar to $BF_{SkipList}^i$, the i^{th} MBR in $MBR_{SkipList}^i$ is the minimum bound rectangle of all the MBRs at each block's head which we represent in formula (3).

$$MBR_{SkipList}^i = MBR_{block}^{current} + \dots + MBR_{block}^{current+\alpha^i} \quad (3)$$

Therefore, the first level's MBR in the skip list can be used to determine whether there are existing transactions that satisfy the query condition in the next α^i blocks.

5.2 Inter-block Query of Discrete-Valued Attributes

For the inter-block query of discrete-valued attributes, we can use bloom-filters in the skip list to help us query quicker. In this way, we also save the time of traversing unnecessary bloom-filters at the head of each block. The specific query algorithm is shown in Algorithm 2. This can be illustrated by the fact that if $\alpha = 2$, we will set the first block of the blockchain as the current block and query the first level's bloom filter in the skip list of the current block. If the return result is true, we will query the bloom-filters $BF_{block}^2, BF_{block}^3$ at the head of the next two blocks. If false is returned, the second level's bloom filter in the skip list of the current block is checked. Eventually, when the $SkipList_{BF}^i$ returns true, it proves that there are might existing blocks that satisfy the query condition between the $(2^{i-1})^{th}$ block or $(2^i)^{th}$ block. Then we need to set the current block as the $(2^{i-1})^{th}$ block and follow the steps above to continue the query process.

5.3 Inter-block Query of Continuous-Valued Attributes

Similar to the inter-block query of discrete-valued attributes, we use MBR at the head of each block to generate the $SkipList_{MBR}$ and propose an efficient scheme for the inter-block query of continuous-valued attributes. We set the current block as the first one which denotes as $block_{current}$. Querying process is started from the first level in the skip list of the first block. If the returned result is true, we need to check the MBRs of the next two blocks. If false is

Algorithm 2 InterQuery($block_{current}, Q$)

Input: Query condition $q = \langle Q_{discrete}, MBR_{query} \rangle$

```

1: for  $BF_i$  and  $MBR_i$  in  $Skiplist$  do
2:   if  $BF_i.isContain()$  AND  $MBR_i.isIntersect()$  then
3:     if  $i \neq 0$  then
4:       InterQuery( $block_{current+\alpha^i}, Q$ )
5:     else
6:       TraverseBlock( $block_{current}, block_{current+\alpha^i}$ )
7:     end if
8:   end if
9: end for
```

returned, the second MBR and MBR behind in the skip list is queried. There are some transactions in the blocks between $(\alpha^{i-1})^{th}$ and $(\alpha^i)^{th}$ satisfying the query conditions if the check result of the $SkipList_{MBR}^i$ returns true. Then we set the $(\alpha^{i-1})^{th}$ block as the current block and continue to query the rest blocks as described above.

6 Performance Analysis

6.1 Analysis of the Efficiency

We analyze the efficiency of our proposed query schemes on the blockchain. Firstly, the intra-block query cost for multi-dimensional continuous-valued attributes is similar to R-tree [18]. We use d_f and d_l to denote the average fan-out of the leaf nodes and the internal nodes in each blocks' R-Tree respectively. In each block, if the number of transactions is N_{block} , the number of leaf nodes and internal nodes in this R-Tree is $\frac{N_{block}}{d_f}$ and $\frac{N_{block}}{d_l}$ [19]. In the unit space $[0, 1]^d$ which contains d dimensions, the probability of two rectangles R_1 and R_2 overlap is as follows in Eq. (4), where R_l^i express the rectangle R's length along the i^{th} dimension [20].

$$P_{overlap} = \prod_{i=1}^d (R_{1,l}^i + R_{2,l}^i) \tag{4}$$

We assume that the total sample space is $[0, s^{\frac{1}{d}}]^d$ and the size of each leaf nodes are the same which we denote as $S_1 = S_2 = \dots = S_n$, so the size of each leaf node equals to $\frac{s \cdot d_f}{N_{block}}$. Similarly, the size of each internal node in level j is $\frac{s}{d_l^j}$. If the length of query condition for continuous-valued attributes in each dimension is $Q_r^{l,i}$ and the length for each node in MerkleRB-Tree in every dimension is the same, then the number of nodes in each block's MerkleRB-Tree that needs to be accessed can be computed like in Eq. (5).

$$N_q = \prod_{i=1}^d \left(\sqrt[d]{\frac{s \cdot d_f}{N_{block}}} + Q_r^{l,i} \right) \cdot d_f + \sum_{j=0}^{h-2} d_l^j \cdot \prod_{i=1}^d \left(\sqrt[d]{\frac{s}{d_l^j}} + Q_r^{l,i} \right) \tag{5}$$

where the height of the MerkleRB-Tree is $h = 1 + \log_{d_l} \frac{s \cdot d_f}{N_{block}}$. Therefore, the total average cost of continuous-valued attributes' query in each block is C_{range} , and the cost of querying continuous-valued attributes each node in MerkleRB-Tree is C_{access} . We illustrate it in Eq. (6).

$$C_{range} = C_{access} \cdot N_q \tag{6}$$

For discrete-valued attributes, we assume that the average probability of a bloom filter contains the discrete query condition value Q_{dis} in each block can be shown in Eq. (7), where the notation θ expresses the number of times that

Q_{dis} appears in the current block. We represent the cost of querying for discrete-valued attributes in Eq. (8).

$$P_{BF}(BF, Q_{dis}) = \frac{\theta}{N_{block}} \quad (7)$$

$$C_{dis} = C_{BF} \cdot \left(\frac{d_f}{N_{block}} + \sum_{j=0}^{h-2} d_l^j \cdot \frac{d_f \cdot f_n^{h-2-j}}{N_{block}} \right) \quad (8)$$

When the query condition contains the continuous-valued attributes and discrete-valued attributes together, it is obvious that total cost for query condition contains both discrete-valued attributes and continuous-valued attributes equals to the right hand of the Eq. (9), in which C_{access} denotes the cost of accessing a node in MBR-Tree.

$$C_{total} = C_{access} \cdot \frac{d_f}{N_{block}} \cdot \prod_{i=1}^d \left(\sqrt[d]{\frac{s d_f}{N_{block}}} + Q_r^{l,i} \right) + \sum_{j=0}^{h-2} d_l^j \cdot \prod_{i=1}^d \left(\sqrt[d]{\frac{s}{d_l^j}} + Q_r^{l,i} \right) \cdot \frac{d_f \cdot f_n^{h-2-j}}{N_{block}} \quad (9)$$

For the inter-block query, we use a skip list to decrease the query cost for both discrete-valued attributes and continuous-valued attributes. The time complexity of the skip list query is $O(\log(n))$ [21]. However, the cost of improving query efficiency is increasing its spatial complexity. And the relationship between time complexity and space complexity in the inter-block query scheme we propose is a negative correlation according to the settings of α and this will be discussed in the next section.

7 Experiments

In this part, we implement and test the performance of the inter-block query and intra-block query respectively.

7.1 Experiment Setting

Dataset. We use a public dataset from kaggle¹. In this dataset, it contains different employees with multi-dimensional attributes including continuous value attributes and discrete value attributes. For this experiment, we choose the year of joining company and the age to do the multi-dimensional range query, and choose city as the discrete value. So for each transaction, we can use $Q = \langle year, age, city \rangle$ to represent the query condition.

Environments. All the experiments are running a computer which is equipped with Intel Core i7 CPU with 6 cores, 3.2GHz for each core. The memory of the computer is 16GB memory on Window 10 operating system. And the JDK version we use is JDK 1.8.

¹ <https://www.kaggle.com/tejashvi14/employee-future-prediction>.

7.2 Performance Evaluation

Query for Discrete-Value Attributes: To verify the efficiency of the query method we propose above, we test the inter-block and intra-block query performance for the discrete-valued attributes. The results are shown in Fig. 3(a) and Fig. 3(b).

In Fig. 3(a), we test our proposed inter-block query schemes on blockchain with different numbers of transactions from 3400 to 4400. We also put the different number of transactions 10, 20 and 40 in each block. We choose the scheme without *SkipList_{BF}* as the baseline. We can see that the query time of our schemes are less than the baseline scheme when each block stores the same amount of data. This is because the fact that the method we propose saves time for querying unnecessary blocks when using *SkipList_{BF}*. In addition, with the increasing number of blocks, the efficiency of our plan is more obvious.

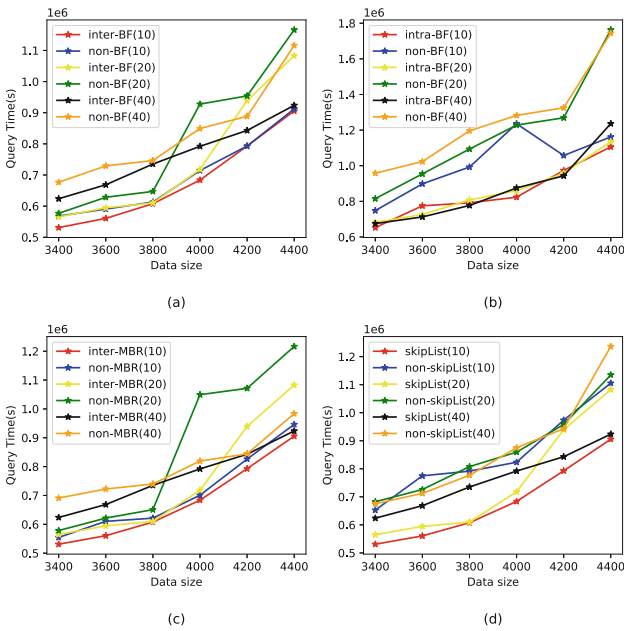


Fig. 3. Query performance

In Fig. 3(b), we put different amounts of data 10, 20 and 40 in each block. As for the circumstance which does not have bloom-filters, we cannot quickly exclude subtrees that do not need to be traversed in MBR-Tree through the discrete-valued query condition. Thus, we need to traverse the nodes that only satisfy the continuous-valued query condition and return the correct nodes. By comparing our proposed intra-block query method for the discrete-value attribute with the baseline scheme without bloom-filters, when the amount of

data inside the block is the same, it can be seen that the performance of our scheme is better than the baseline scheme. As the amount of data inside the block increases, the query cost of our solution increases smoothly, whereas the query cost of the baseline method increases apparently. The reason lies in that as the number of nodes increases, the baseline method needs to query more useless subtrees, and the method we propose can solve this type of problem effectively.

Query for Continuous-Value Attributes: For query continuous-valued attributes, it is obvious that the intra-block query performance of our scheme using R-Tree is much better than non-index query scheme [7]. Moreover, the performance results of continuous-valued inter-block query efficiency for multi-dimensional data are shown in Figs. 3(c). We choose the method without $SkipList_{MBR}$ as the baseline method. We can see that our scheme performs better than the Baseline scheme, since our proposed scheme for inter-block queries saves the time cost to search unnecessary blocks in blockchain by using $SkipList_{MBR}$.

In Fig. 3(d), we contrast the method without skip list for inter-block query process to our scheme. It can demonstrate the advantages of generating and applying skip list for the inter-block query process in blockchain-based FL.

8 Conclusion

In this paper, we optimize the query efficiency of selecting participants in blockchain-based FL by modifying the blockchain's structure. By analyzing and comparing the existing query schemes, our scheme that contains intra-block query and inter-block query has superiority on query performance. In the future, we will further do explorations in industrial platform of blockchain for varies fields.

Acknowledgement. This work is supported by the Sichuan Provincial Key Research and Development Program (2020YFQ0056, 2021ZHCG0001, 2021YFG0132, 2021GFW046, 2022YFSY0005, 22ZDZX0046).

References

1. Yang, Q., Liu, Y., Chen, T., Tong, Y.: Federated machine learning: concept and applications. *ACM Trans. Intell. Syst. Technol. (TIST)* **10**(2), 1–19 (2019)
2. Kim, H., Park, J., Bennis, M., Kim, S.-L.: Blockchain on-device federated learning. *IEEE Commun. Lett.* **24**(6), 1279–1283 (2019)
3. Warnat-Herresthal, S., et al.: Swarm learning for decentralized and confidential clinical machine learning. *Nature* **594**(7862), 265–270 (2021)
4. Zhang, Q., Palacharla, P., Sekiya, M.: A blockchain based protocol for federated learning. In: 2020 IEEE 28th International Conference on Network Protocols (ICNP), pp. 1–2. IEEE (2020)
5. Kumar, R., et al.: Blockchain-federated-learning and deep learning models for COVID-19 detection using CT imaging. *IEEE Sens. J.* **21**(14), 16301–16314 (2021)

6. Peng, Z., Wu, H., Xiao, B., Guo, S.: VQL: providing query efficiency and data authenticity in blockchain systems. In: 2019 IEEE 35th International Conference on Data Engineering Workshops (ICDEW), pp. 1–6. IEEE (2019)
7. Theodoridis, Y., Vazirgiannis, M., Sellis, T.: Spatio-temporal indexing for large multimedia applications. In: Proceedings of the Third IEEE International Conference on Multimedia Computing and Systems, pp. 441–448. IEEE (1996)
8. Zhang, C., Xu, C., Xu, J., Tang, Y., Choi, B.: Gem²-tree: a gas-efficient structure for authenticated range queries in blockchain. In: 2019 IEEE 35th International Conference on Data Engineering (ICDE), pp. 842–853. IEEE (2019)
9. Kairouz, P., et al.: Advances and open problems in federated learning, arXiv preprint [arXiv:1912.04977](https://arxiv.org/abs/1912.04977) (2019)
10. Billah, M., Mehedi, S., Anwar, A., Rahman, Z., Islam, R., et al.: A systematic literature review on blockchain enabled federated learning framework for internet of vehicles, arXiv preprint [arXiv:2203.05192](https://arxiv.org/abs/2203.05192) (2022)
11. Kim, H., Park, J., Bennis, M., Kim, S.L.: Blockchain on-device federated learning. *IEEE Commun. Lett.* **24**(6), 1279–1283 (2020)
12. Nathan, S., Govindarajan, C., Saraf, A., Sethi, M., Jayachandran, P.: Blockchain meets database: design and implementation of a blockchain relational database. *Proc. VLDB Endow.* (2019)
13. Muzammal, M., Qu, Q., Nasrulin, B., et al.: ChainSQL: a blockchain database application platform, arXiv preprint (2019)
14. Li, Y., Zheng, K., Yan, Y., Liu, Q., Zhou, X.: EtherQL: a query layer for blockchain system. In: Candan, S., Chen, L., Pedersen, T.B., Chang, L., Hua, W. (eds.) DAS-FAA 2017. LNCS, vol. 10178, pp. 556–567. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-55699-4_34
15. Yang, Y., Papadopoulos, S., Papadias, D., Kollios, G.: Spatial outsourcing for location-based services. In: 2008 IEEE 24th International Conference on Data Engineering, pp. 1082–1091. IEEE (2008)
16. Kamel, I., Faloutsos, C.: Hilbert R-tree: an improved R-tree using fractals. Technical report (1993)
17. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. *Commun. ACM* **13**(7), 422–426 (1970)
18. Yang, Y., Papadopoulos, S., Papadias, D., Kollios, G.: Authenticated indexing for outsourced spatial databases. *VLDB J.* **18**(3), 631–648 (2009)
19. Theodoridis, Y., Sellis, T.: A model for the prediction of R-tree performance. In: Proceedings of the Fifteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pp. 161–171 (1996)
20. Pagel, B.-U., Six, H.-W., Toben, H., Widmayer, P.: Towards an analysis of range query performance in spatial data structures. In: Proceedings of the Twelfth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pp. 214–221 (1993)
21. Pugh, W.: Skip lists: a probabilistic alternative to balanced trees. *Commun. ACM* **33**(6), 668–676 (1990)



Joint Edge Server Deployment and Service Placement for Edge Computing-Enabled Maritime Internet of Things

Chaoyue Zhang¹, Bin Lin^{1,2(✉)}, Lin X. Cai³, Liping Qian⁴, Yuan Wu⁵,
and Shuang Qi¹

¹ School of Information Science and Technology, Dalian Maritime University,
Dalian 116026, China
binlin@dlmu.edu.cn

² Peng Cheng Laboratory, Shenzhen 518052, China

³ Department of Electrical and Computer Engineering,
Illinois Institute of Technology, Chicago, USA

⁴ College of Information Engineering, Zhejiang University of Technology,
Hangzhou 310014, China

⁵ State Key Laboratory of Internet of Things for Smart City, University of Macau,
Macau, China

Abstract. With the growing activities of diverse Maritime Internet of Things (MIoT), mobile edge computing (MEC) becomes a promising paradigm to provision computation and storage for computation-intensive tasks of marine users. Although the edge server (ES) deployment and service placement are important issues in the field of MEC, research on joint placement is often overlooked, particularly in the MIoT. In this paper, we propose the buoy-based ES deployment and service placement (BESDSP) problem for MIoT networks, aiming at maximizing the total profit while considering the location constraints of buoys, the different service request rates, the income and delay cost of service provided by ESs, as well as the characteristics of maritime channels. Then, we propose a heuristic approach, the genetic-BESDSP (G-BESDSP) algorithm, to solve the BESDSP problem. Simulation results demonstrate that the proposed G-BESDSP algorithm outperforms existing state of art solutions.

Keywords: Mobile edge computing (MEC) · Edge server (ES)
deployment · Service placement · Maritime Internet of Things (MIoT)

1 Introduction

With the rapid development of artificial intelligence and communication technologies, a variety of delay sensitive and computation intensive maritime services

such as virtual/augmented reality, automatic driving and intelligent transportation have involved in the field of maritime services, and Maritime Internet of Things (MIoT) have been emerging. In the traditional cloud computing architecture, mobile users offload task requests to a remote cloud server which is inefficient due to low bandwidth and high latency. A massive amount of maritime data transmission in MIoT puts forward higher requirements for bandwidth. Furthermore, mobile edge computing (MEC) as one core component of 5G networks, brings a promising paradigm for deploying servers at the infrastructure-based edge of networks in proximity to the users of MIoT. MEC migrates a portion of communication, computing and storage resources from the remote cloud server to the edge, in order to reduce the end-to-end transmission delay and decrease the traffic of the backbone network of MIoT.

Most existing MEC works focus on task offloading [1] and resource allocation [2] strategies for users, assuming that all edge servers (ESs) have been deployed. In terms of ES deployment, recent studies assume that each ES places a kind of service, rarely taking the varieties of services and users' requests for different services into account [3–8]. In terms of service placement, recent studies assume that all base-stations are MEC-enabled [9–12]. In MEC networks, ES deployment and service placement are tightly coupled, which should be joint optimized. Moreover, research on MEC in maritime networks is lagging behind that in the terrestrial networks. The challenges of maritime networks are summarized as follows. i) The hostile maritime environment makes it challenging to reliably deliver data over a communication link. ii) Different from terrestrial networks, the deployment of maritime networks are subject to many constraints as it is difficult to find a solid area to install network devices. Specially, the buoy provides a feasible solution to establish wireless networks. ESs can be deployed on the existing buoys in MIoT. Thus, the physical locations of buoys are important as ESs installed on buoys process different service requests from users. As mentioned above, we need to consider joint ES deployment and service placement [3, 13, 14], which is a crucial issue in the field of MEC. However, no matter disjoint or joint ES deployment and service placement, few works pay attention to maritime MEC environments. From the perspective of marine users, the reasonable ES deployment and service placement scheme effectively reduces the completion time of the offloading task. From the perspective of service providers, although a large number of deployed ESs can improve the edge resources, it will result in high operational costs, such as server device prices, installation and maintenance costs. Therefore, deploying ESs at appropriate numbers and locations to maximize the profit of service providers and to provide satisfactory QoS for users is an important issue.

In this paper, we study a buoy-based ES deployment and service placement (BESDSP) problem, aiming at maximizing the total profit of all ESs. Specially, we first determine a set of appropriate buoys for ESs deployment, based on which the services are then placed at each ES. The main contributions are summarized below:

- We propose a novel deployment model in an MEC-enabled MIoT network. Then, we formulate a BESDSP problem with the goal of maximizing the total profit, taking into consideration the location constraints of buoys, the user’s request rate, the income and delay cost of service, the storage capacity and the computing capacity of the ES, as well as the characteristics of maritime channels.
- We propose a genetic-BESDSP (G-BESDSP) algorithm, to solve the BESDSP problem, and compare it with some state of art algorithms in the literature.

The rest of this paper is organized as follows: Sect. 2 formulates the BESDSP problem. Section 3 describes the proposed G-BESDSP algorithm in detail. Section 4 evaluates the performance of G-BESDSP algorithm and compares it with existing representative algorithms. Section 5 concludes this paper.

2 System Model and Problem Formulation

2.1 System Model

We consider an MEC-enabled MIoT network as shown in Fig. 1. Buoys are employed as micro base stations that receive marine users’ service requests and then forward the service requests to ESs.

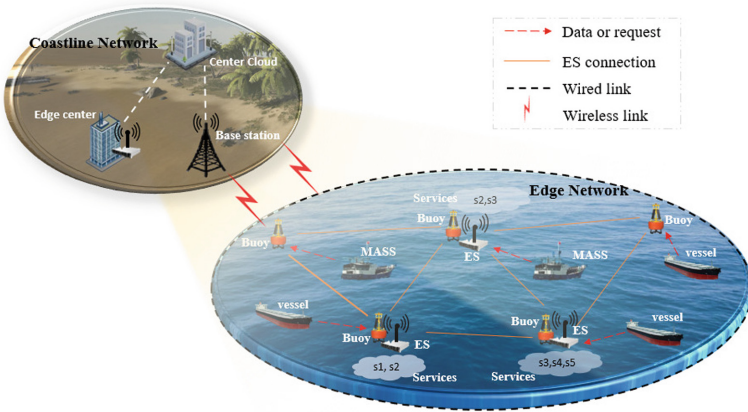


Fig. 1. MEC-enabled MIoT network system model.

The MEC-enabled MIoT network can be described as an undirected graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{V_1, V_2, \dots, V_m, \dots, V_M\}$ is the set containing M buoys and \mathcal{E} is the set of communication links between these buoys. Each marine user $w \in \mathcal{W} \triangleq \{1, 2, \dots, W\}$ can access the buoy and offload tasks to the ES. There are K ESs denoted as $\mathcal{K} = \{1, 2, \dots, K\}$. We assume that the number of existing buoys is greater than that of the ESs K ($M > K$) to be deployed. We suppose

that each buoy attaches only one ES; and only one ES can be installed in each buoy. Then, a binary variable $y_{V_m} \in \{0, 1\}, \forall V_m \in \mathcal{V}$ is denoted as whether buoy V_m hosts an ES or not, where $y_{V_m} = 1$ indicates that the ES is deployed on the buoy V_m , and $y_{V_m} = 0$ otherwise. Therefore, we have

$$\sum_{V_m=1}^M y_{V_m} = K \quad (1)$$

Each service $s \in \mathcal{S} \triangleq \{1, 2, \dots, S\}$ is modeled as a tetrad $\mathcal{D}_s = \{\alpha_s, g_s, q_s, \beta_s\}$, where α_s denotes the average request rate of service s for each user, g_s denotes the storage size of service s , q_s denotes the unit income of providing the service s , and β_s denotes the delay cost of processing the service s . Binary variable $x_{V_m}^n \in \{0, 1\}, \forall V_m \in \mathcal{V}, \forall n \in \mathcal{K}$ is defined to indicate whether ES n is responsible for the buoy V_m hosting its service requests ($x_{V_m}^n = 1$) or not ($x_{V_m}^n = 0$). Each ES can be responsible for several buoys and each buoy attaches only one ES, which is formulated as follows,

$$\sum_{n=1}^K x_{V_m}^n = 1, \forall V_m \in \mathcal{V} \quad (2)$$

The buoy serves as a relay node to forward the user's service request to the corresponding ES. Each user can be covered by several buoys, but a user can only communicate with a buoy at one time. The total number of users served by ES n is expressed as

$$U_n = \sum_{V_m=1}^M u_{V_m} x_{V_m}^n \quad (3)$$

where u_{V_m} represents the number of users connected to the buoy V_m . In order to satisfy marine users' service requirements, ESs usually provision multiple difference services. We use a binary variable $b_n^s \in \{0, 1\}, \forall n \in \mathcal{K}, \forall s \in \mathcal{S}$ to denote the placement of services, where $b_n^s = 1$ indicates service s is placed on ES n , and $b_n^s = 0$ otherwise. Furthermore, we denote $\mathcal{B}_n = \{s \mid b_n^s = 1, s \in \mathcal{S}\}$ by the set of ES n placed services, then the users' service requests of ES n is expressed as $Req_n = \sum_{s \in \mathcal{B}_n} U_n \alpha_s, \forall n \in \mathcal{K}$. The limited storage size of each ES is G , which is used to store services. We consider the storage size of service s , the total storage size of a series of services placed on each ES cannot exceed the storage size of the ES, the constraint can be expressed as follows,

$$\sum_{s \in \mathcal{B}_n} g_s \leq G, \forall n \in \mathcal{K} \quad (4)$$

where g_s is the storage size of service s . From the perspective of service providers, the payment of edge users is necessary, and once the service request is processed will generate corresponding revenue which is different for each service. From the perspective of marine users, the offloaded task processing latency at the ES is the major expenditure on service, which is considered as the server penalty. In

summary, the objective function of the BESDSP problem is composed of income and delay cost.

- 1) *Income of ESs*: The income of ESs is related to the placed services and connected users. The income of providing service requests of ES n is modeled as follows,

$$Q_n = \sum_{s \in B_n} U_n \alpha_s q_s, \forall n \in \mathcal{K} \quad (5)$$

where q_s is the unit income of providing service s , α_s is the average request rate of service s for each user, and U_n is the total number of users served by ES n .

- 2) *Delay Cost of Services*: In this paper, considering the impacts of sea surface reflection and antenna height, the two-ray signal propagation model is adopted for maritime channels. The transmission model assumes that the maritime channel mainly consists of a direct path and a reflection path, and its path loss can be expressed as [15]

$$L_{2\text{-ray}} = -10 \log_{10} \left\{ \left(\frac{\lambda}{4\pi d} \right)^2 \left[2 \sin \left(\frac{2\pi H_1 H_2}{\lambda d} \right) \right]^2 \right\} \quad (6)$$

where λ is the carrier wavelength, d is the distance between the transmitter and receiver, H_1 is transmitter antenna height, H_2 is receiver antenna height. Using the two-ray signal model, the channel gain from user w to ES n can be expressed as

$$h_{w,n} = \left(\frac{\lambda}{4\pi d_{w,n}} \right)^2 \left[2 \sin \left(\frac{2\pi H_w H_n}{\lambda d_{w,n}} \right) \right]^2, \forall n \in \mathcal{K}, \forall w \in \mathcal{W} \quad (7)$$

where $d_{w,n}$ is the direct signal link distance between user w and ES n , H_w is the user w antenna height, H_n is the ES n antenna height. We can express the maximum achievable rate from user w to ES n as

$$\begin{aligned} \sigma_{w,n} &= B \log_2 \left(1 + \frac{P_w h_{w,n}}{N_0} \right) \\ &= B \log_2 \left[1 + \frac{P_w}{N_0} \left(\frac{\lambda}{4\pi d_{w,n}} \right)^2 \left[2 \sin \left(\frac{2\pi H_w H_n}{\lambda d_{w,n}} \right) \right]^2 \right], \forall n \in \mathcal{K}, \forall w \in \mathcal{W} \end{aligned} \quad (8)$$

where N_0 denotes the white Gaussian noise power density, B denotes available bandwidth, P_w denotes the user w transmission power.

The delay cost is related to the end-to-end latency in providing the request. Assume that each ES handles service requests with first input first output method. When the user requests service to the ES, the task will be stored in the queue, if the server is busy. The delay is the sum of the transmission delay, the processing delay and the queuing delay. Therefore, the delay of task e on the ES

n can be expressed as [8]

$$T(ES_n, e) = \begin{cases} \frac{\theta_e}{\sigma_{w,n}} + \frac{\theta_e}{\mu} & e = 1 \\ \max \left\{ T(ES_n, e - 1), \frac{\theta_e}{\sigma_{w,n}} \right\} + \frac{\theta_e}{\mu} & 2 \leq e \leq \lceil Req_n \rceil \end{cases} \quad (9)$$

where $\lceil \cdot \rceil$ is the integer upward, θ_e is the data size of task e processed by ES n , $\sigma_{w,n}$ is the maximum achievable rate from user w to ES n , μ is the computation capacity of ES. We use a binary variable $\zeta_{n,e}^s \in \{0, 1\}$ to denote whether task e belongs to service $s \in \mathcal{B}_n$ processed by ES n ($\zeta_{n,e}^s = 1$) or not ($\zeta_{n,e}^s = 0$). The delay cost of services request to ES n is calculated as follows,

$$cost_n = \sum_{s \in \mathcal{B}_n} \sum_{e=1}^{\lceil Req_n \rceil} \zeta_{n,e}^s T(ES_n, e) \beta_s, \forall n \in \mathcal{K} \quad (10)$$

Thus, the profit of ES n is computed as follows,

$$\begin{aligned} P_n &= Q_n - \rho cost_n \\ &= \sum_{s \in \mathcal{B}_n} U_n \alpha_s q_s - \rho \sum_{s \in \mathcal{B}_n} \sum_{e=1}^{\lceil Req_n \rceil} \zeta_{n,e}^s T(ES_n, e) \beta_s \end{aligned} \quad (11)$$

where $\rho > 0$ is a parameter defining the relative weight of income and delay cost.

2.2 Problem Formulation

In this subsection, we aim to maximize the total profit via optimizing the income of all ESs and the delay cost of services processed by ESs to find a feasible deployment strategy. Thus, the BESDSP problem is formulated as

$$\begin{aligned} \max \quad & \sum_{n=1}^K P_n \\ \text{s.t. C1:} \quad & \sum_{n=1}^K x_{V_m}^n = 1, \forall V_m \in \mathcal{V} \\ \text{C2:} \quad & b_n^s \in \{0, 1\}, \forall n \in \mathcal{K}, \forall s \in \mathcal{S} \\ \text{C3:} \quad & \sum_{s \in \mathcal{B}_n} g_s \leq G, \forall n \in \mathcal{K} \\ \text{C4:} \quad & y_{V_m} \in \{0, 1\}, \forall V_m \in \mathcal{V} \\ \text{C5:} \quad & \sum_{V_m=1}^M y_{V_m} = K \end{aligned} \quad (12)$$

The constraint C1 ensures that each buoy is associated with only one ES. The constraint C2 denotes whether service s is placed on ES n ($b_n^s = 1$) or

not ($b_n^s = 0$). The constraint C3 guarantees that the total storage size of placed services cannot exceed the storage capacity of the ES. The constraint C4 denotes whether buoy V_m host an ES ($y_{V_m} = 1$) or not ($y_{V_m} = 0$). The constraint C5 indicates the maximum number of ESs that can be deployed.

The BESDSP problem is NP-hard. We analyse the NP-hardness of the BESDSP problem by a reduction from the warehouse-retailer network design (WRND) problem. We model our problem to the WRND problem in the following way: i) The optimal location of opened warehouses in the WRND problem is mapped to the appropriate buoys to deploy ESs in the BESDSP problem. ii) Each retailer served by only one opened warehouse in the WRND problem is mapped to each marine user served by only one ES-enabled buoy. iii) The inventory replenished by a supplier in the WRND problem is mapped to service instances placed on each ES in the BESDSP problem. iv) The cost of the WRND problem is mapped to the delay cost of the BESDSP problem. The WRND problem is known to be NP-hard [16], and hence the BESDSP problem is NP-hard.

3 Genetic-BESDSP Algorithm

In this section, we propose a genetic-BESDSP (G-BESDSP) algorithm to solve the formulated BESDSP problem.

The decision variable of the BESDSP problem is encoded into a string of alphabets [7]. These strings are referred to as chromosomes. A chromosome contains two decision variables, ES deployment y_{V_m} and service placement b_n^s . An example of constructed chromosome is shown in Fig. 2, the chromosome $D = \{X_1, X_2\}$ represents the feasible solution to the problem that consists of optimization variables sub-chromosome X_1 and sub-chromosome X_2 . Each binary coded gene in a sub-chromosome X_1 is used to present the buoy location, which is decided by the number of buoys m . Each binary coded gene in a sub-chromosome X_2 is applied to represent the ES which the number of ESs is k . In addition, binary encoding of the gene in X_2 is used to present a service instance which the number of services is n .

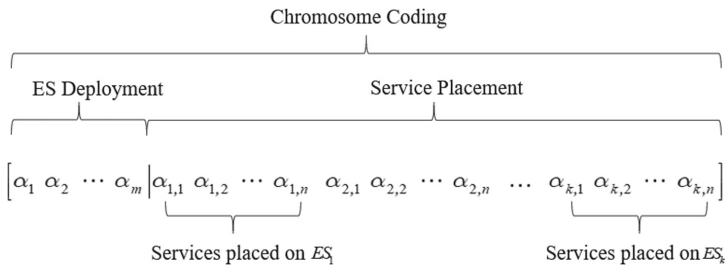


Fig. 2. An example of constructed chromosome.

- 1) *Initialization and Fitness Function*: In this phase, an initial population with a size of P is formed from different combinations of chromosomes. The fitness function is used to evaluate the individual chromosome in the population in each generation of the evolution. According to the theory of survival of the fittest, the higher value of individual fitness, the less likely it is to be eliminated. Therefore, the fitness function $F(D_x)$ is dependent on the objective function.
- 2) *Elitist Preservation Strategy*: Initial population chromosomes are evaluated by fitness function for individual selection operation. Elitist preservation strategy is used to retain R individuals with high fitness scores from the parent generation into the children generation, which can effectively prevent the existence of losing high-quality genes. The chromosome with the high fitness score will be chosen as the next generation population, which prevents the crossover and mutation operations from losing elite individuals.
- 3) *Crossover*: To avoid the emergence of premature convergence, individuals are carried out crossover and mutation operations. The crossover strategy is to select two individuals from the parent generation, and swap parts of individuals to produce two new individuals into the next generation. The crossover process is described as follows: *i*) Select two individuals D_x and D_y as the parent chromosomes for the crossover operation. *ii*) Retain the characteristics of the constructed chromosome, while swapping parts of D_x and D_y . *iii*) New chromosomes generated by cross operation are added to the next generation of the population.
- 4) *Mutation*: The mutation operation slightly alters the genes of individuals except for the elite. The detailed process of mutation operation is as follows: *i*) If the mutation probability is satisfied then the selected individual performs the mutation operation, otherwise the individual remains unchanged. *ii*) The mutation operator randomly selects the genes of sub-chromosome X_1 , retaining the number of ESs is necessary. If the gene value of the selected gene is 1, it becomes 0 after the mutation and vice versa. *iii*) The mutation operator randomly selects the genes of sub-chromosome X_2 , while retaining the capacity constraints for each ES.
- 5) *Roulette Selection Strategy*: The remaining individuals are selected through the roulette selection strategy and copied directly to the next generation. In this phase, each individual's selected probability $L(D_x)$ is relevant to its fitness value, that is

$$L(D_x) = \frac{F(D_x)}{\sum_{x=1}^P F(D_x)} \quad (13)$$

After calculating the selected probability of all individuals, the individual's cumulative probability is getting as follows,

$$Q(D_x) = \sum_{y=1}^x L(D_y) \quad (14)$$

The individual's cumulative probability is the sum of the selected probability of all individuals previously selected by the individual, which is equivalent

to the probability distribution function in probability theory. Roulette selection strategy is described as follows: *i*) Calculate the individual's selected probability $L(D_x)$ according to its fitness value. *ii*) Calculate the individual's cumulative probability $Q(D_x)$. *iii*) Generate a random number $c \in [0, 1]$, the selected individual $Q(D_p)$ is shown as

$$Q(D_p) = \begin{cases} c \leq Q(D_1), & p = 1 \\ Q(D_{p-1}) \leq c \leq Q(D_p), & 2 \leq p \leq P \end{cases} \quad (15)$$

iv) Repeat step. *iii*) Until all requirements are met. The pseudocode of the G-BESDSP algorithm is presented in Algorithm 1.

Algorithm 1 Genetic-BESDSP (G-BESDSP) Algorithm

Input:

The set of buoys \mathcal{V}
 The set of users \mathcal{W}
 The locations of buoys \mathcal{V}_{xy}
 The locations of users \mathcal{W}_{xy}
 The set of services \mathcal{S} and corresponding parameters \mathcal{D}_s
 The storage size of edge server G
 The population size Z
 The maximum number of iterations I

Output:

A joint edge deployment and service placement decision Λ^{opt}

```

1: Initialize Population  $\mathcal{P}$ 
2:  $ge \leftarrow 0$ 
3: while  $ge < I$  do
4:   edge server deployment and service placement scheme  $\Lambda^z$ 
5:   for  $z$  in  $Z$  do
6:     edge server deployment and service placement scheme  $\Lambda^z$ 
7:     for  $w$ -th user request service  $w_s$  in  $\mathcal{S}$  do
8:       if  $w_s$  is in service placed by the nearest edge server  $k$  forwarded by the
9:          $w$ -th user's subordinate buoy then
10:          task  $w_s$  transmits to the  $k$ -th edge server
11:        else
12:          task  $w_s$  transmits to the nearby edge server with placed service  $s$ 
13:        end if
14:      end for
15:    elite individual
16:    crossover
17:    mutation
18:    roulette selection individual
19:     $ge \leftarrow ge + 1$ 
20:  end while
21: find  $\Lambda^{opt}$  according to the fitness function
22: return  $\Lambda^{opt}$ 
    
```

4 Simulation Results and Discussion

In this section, we conduct simulation experiments to validate the effectiveness of proposed G-BESDSP algorithm. Simulation experiments coded with Python 3.9. To verify the performance of the proposed algorithm, we compare it with a randomized ES deployment and randomized service placement (RSDRSP) algorithm, as well as a cluster placement (CP) algorithm. RSDRSP algorithm selects buoys randomly to deploy ESs and then places services randomly for each ES. By dividing buoys into K parts, CP algorithm selects the buoy closest to the

centroid of each part as the position of the ES [4], and then adopts our proposed service placement method to place services for each ES. The simulation scenario is set as a narrow lane of 30 nmile \times 4 nmile in an offshore area (the selected simulation scenario from the sea chart), where marine users and buoys are unevenly distributed. According to the scale of offshore scenario and navigation safety distance limit, we set the number of marine users from 10 to 35. The average request rate α_s of the marine user for different services follows the Zipf distribution [3], and all users offload required tasks to edge networks. The simulation parameter settings are chosen according to [5, 17, 18] and summarized in Table 1.

Table 1. Simulation parameters.

Symbol	Parameter name	Default value
Θ	Scale of offshore scenario	30 nmile \times 4 nmile
M	Number of buoys	10
K	Number of ESs	3,4,5,6,7,8
W	Number of marine users	10, 15, 20, 25, 30, 35
μ	Computation capacity of ES	3.0 GHz
G	Storage size of ES	96 GB
\mathcal{S}	Set of services	{1, 2, 3}
λ	Carrier wavelength	0.05 m
B	Communication bandwidth per link	10 MHz
H_w	Antenna height of user	9.8 m
H_n	Antenna height of buoy	1.9 m
P_w	User transmission power	47 dBm
N_0	White Gaussian noise power density	-169 dBm/Hz

Impact of the Number of ESs: We first study the total profit under different number of ESs, using different algorithms. As shown in Fig. 3, the total profit of the edge network is lowest when the number of deployed ESs is 3. The total profit improves significantly when the number of ESs increases from 3 to 4. Subsequently, the profit does not vary much when the number of ESs increases. The reason is the deployed ESs cannot carry mass service requests from marine users in the edge network, so that ESs with high delay produce excessive latency cost. The edge network has more resources to process offloaded tasks from the marine users with the increase in the number of ESs. Thus, the available resources of the edge network far exceed the demand of users with more ESs deployed that enables low delay. On the contrary, it reduces the resource utilization of ESs and increases the additional equipment cost of ESs. The performance of the service delay of all ESs are compared in Fig. 4. It can be seen that the delay decreases

when the number of ESs increases. This is due to the fact that there are more computation resources to process offloaded tasks. In addition, Fig. 3 and Fig. 4 also show that the G-BESDSP algorithm outperforms the RSDRSP and CP algorithms in terms of the total profit and the delay.

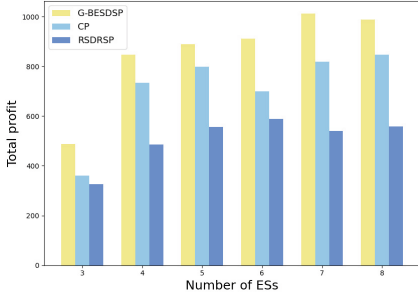


Fig. 3. Comparison of the total profit under different numbers of ESs

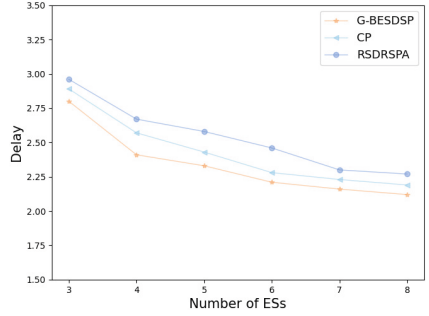


Fig. 4. Comparison of the delay under different numbers of ESs

Impact of the Number of Marine Users: We compare the total profit and delay under different numbers of marine users. In Fig. 5, the total profit increases with the number of marine users in all three algorithms. This is because, when the number of marine users increases, more tasks produced by users need to be processed on ESs, which leads to an increase in the service income. Moreover, the available resources of ESs can meet the demand of marine users, which enables low delay cost. As shown in Fig. 6, the delay does not vary much when the number of marine users from 10 to 20. While the delay of the G-BESDSP algorithm is

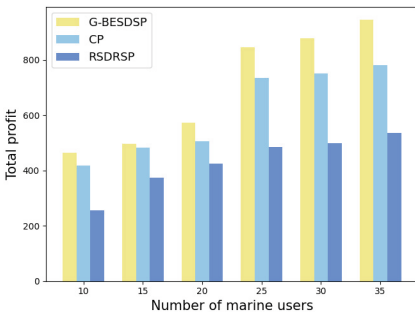


Fig. 5. Comparison of the total profit under different numbers of marine users

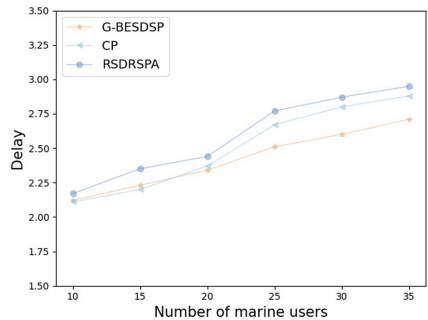


Fig. 6. Comparison of the delay under different numbers of marine users

lower than the RSDRSP and CP algorithms with the number of marine users increases. Hence, the G-BESDSP algorithm can service more marine users and thus achieves the highest total profit while obtaining a low delay.

5 Conclusion

We have formulated a buoy-based ES deployment and service placement problem in the MEC-enabled MIIoT network by maximizing the total profit. The BESDSP problem is NP-hard, hence a genetic algorithm G-BESDSP is proposed. Finally, simulation results have demonstrated that the proposed algorithm achieves close to optimal performance. In the future work, we will explore the computing migration to further improve the mobile users' experience in an MEC-enabled MIIoT network.

Acknowledgements. The work was supported by the National Key Research and Development Program of China (No: 2019YFE0111600), National Natural Science Foundation of China (No: 61971083 and No: 51939001), LiaoNing Revitalization Talents Program (No: XLYC2002078) and the Major Key Project of PCL (PCL2021A03-1).

References

1. Zhang, J., Guo, H., Liu, J., Zhang, Y.: Task offloading in vehicular edge computing networks: a load-balancing solution. *IEEE Trans. Veh. Technol.* **69**(2), 2092–2104 (2020)
2. Qian, L.P., Shi, B., Wu, Y., Sun, B., Tsang, D.H.K.: NOMA-enabled mobile edge computing for internet of things via joint communication and computation resource allocations. *IEEE Internet Things J.* **7**(1), 718–733 (2020)
3. Zhang, X., Li, Z., Lai, C., Zhang, J.: Joint edge server placement and service placement in mobile edge computing. *IEEE Internet Things J.* (Early Access). <https://doi.org/10.1109/JIOT.2021.3125957>
4. Cao, B., et al.: Large-scale many-objective deployment optimization of edge servers. *IEEE Trans. Intell. Transp. Syst.* **22**(6), 3841–3849 (2021)
5. Cao, K., Li, L., Cui, Y., Wei, T., Hu, S.: Exploring placement of heterogeneous edge servers for response time minimization in mobile edge-cloud computing. *IEEE Trans. Industr. Inform.* **17**(1), 494–503 (2021)
6. Chen, X., Liu, W., Chen, J., Zhou, J.: An edge server placement algorithm in edge computing environment. In: 2020 12th International Conference on Advanced Infocomm Technology (ICAIT), Macao, China, pp. 85–89 (2020)
7. Kasi, S.K., et al.: Heuristic edge server placement in industrial internet of things and cellular networks. *IEEE Internet Things J.* **8**(13), 10308–10317 (2021)
8. Zhang, J., Lu, J., Yan, X., Xu, X., Qi, L., Dou, W.: Quantified edge server placement with quantum encoding in internet of vehicles. *IEEE Trans. Intell. Transp. Syst.* (Early Access). <https://doi.org/10.1109/TITS.2021.3116960>
9. Martin, J.P., Kandasamy, A., Chandrasekaran, K.: CREW: cost and reliability aware eagle-whale optimiser for service placement in fog. *Softw. Pract. Exp.* **50**(12), 2337–2360 (2020)

10. Gao, B., Zhou, Z., Liu, F., Xu, F., Li, B.: An online framework for joint network selection and service placement in mobile edge computing. *IEEE Trans. Mob. Comput.* (Early Access). <https://doi.org/10.1109/TMC.2021.3064847>
11. Badri, H., Bahreini, T., Grosu, D., Yang, K.: Energy-aware application placement in mobile edge computing: a stochastic optimization approach. *IEEE Trans. Parallel Distrib. Syst.* **31**(4), 909–922 (2020)
12. Chen, Y., et al.: LOCUS: user-perceived delay-aware service placement and user allocation in MEC environment. *IEEE Trans. Parallel Distrib. Syst.* **33**(7), 1581–1592 (2022)
13. Moubayed, A., Shami, A., Heidari, P., Larabi, A., Brunner, R.: Edge-enabled V2X service placement for intelligent transportation systems. *IEEE Trans. Mob. Comput.* **20**(4), 1380–1392 (2021)
14. Zhao, L., et al.: Joint shareability and interference for multiple edge application deployment in mobile-edge computing environment. *IEEE Internet Things J.* **9**(3), 1762–1774 (2022)
15. Wei, T., Feng, W., Chen, Y.F., Wang, C.X., Ge, N., Lu, J.H.: Hybrid satellite-terrestrial communication networks for the maritime internet of things: key technologies, opportunities, and challenges. *IEEE Internet Things J.* **8**(11), 8910–8934 (2021)
16. Teo, C.P., Jia, S.: Warehouse-retailer network design problem. *Oper. Res.* **52**(3), 396–408 (2004)
17. Yang, T., Kong, L., Zhao, N., Sun, R.: Efficient energy and delay tradeoff for vessel communications in SDN based maritime wireless networks. *IEEE Trans. Intell. Transp. Syst.* **22**(6), 3800–3812 (2021)
18. Reyes-Guerrero, J.C., Bruno, M., Mariscal, L.A., Medouri, A.: Buoy-to-ship experimental measurements over sea at 5.8 GHz near urban environments. In: 2011 11th Mediterranean Microwave Symposium (MMS), Hammamet, pp. 320–324 (2011)



Optimal Task Offloading Strategy in Vehicular Edge Computing Based on Game Theory

Zheng Zhang, Lin Wu, and Feng Zeng^(✉)

School of Computer Science and Engineering, Central South University, Changsha,
China

fengzeng@csu.edu.cn

Abstract. In vehicular edge computing, when there are many vehicles requesting offloading services at the same time, relying only on the resources of edge servers often cannot meet the needs of delay-sensitive tasks. Most existing task offloading studies tend to only consider pure offloading strategies for vehicles, which may not be the optimal strategy for some splittable tasks. In this paper, we jointly optimize the vehicle hybrid offloading strategy and the server resource pricing strategy. For a requesting task, it can be executed locally, be offloaded to the edge server, and be offloaded to the cloud center at the same time. We model the interaction between vehicles, the edge server and the cloud center as a game model. Based on the analysis of backward induction, we prove that the game has a unique Nash equilibrium. Meanwhile, an algorithm that can converge to the equilibrium point in polynomial time is proposed. Numerical experimental results show that the proposed algorithm has better performance in terms of delay and cost than existing algorithms.

Keywords: Vehicular edge computing · Task offloading · Game theory · Backward induction

1 Introduction

Currently, more and more vehicular intelligent applications require large amount of computing resources. However, due to vehicle's resource limit, the large amount of computation cannot be completed in the vehicle locally [1]. In addition, the cloud server is far away from the vehicle, which leads to the high latency of data transmission. With the servers deployed nearby the vehicles, the edge network is formed to provide high-intensity computing service for vehicles, and the close distance between vehicles and servers ensures the low-latency data transmission in vehicular edge computing (VEC) [2].

Although edge servers can provide high quality service for requesting vehicles, edge servers have limit resources [3] compared with the cloud center. Therefore, offloading some parts of one task to the cloud center not only can improve

the quality of service (QoS) but also can reduce the load of the edge server. In addition, the vehicle needs to pay the corresponding fee for the services provided by edge and cloud servers. Vehicles hope to obtain high-quality services at low cost, while edge servers hope to maximize their economic benefits. Therefore, reasonable vehicle offloading strategy and server pricing strategy are particularly important in task offloading. But, most existing works do not consider cloud-edge-vehicle tripartite collaborative in task offloading [4–7]. In this paper, the optimal vehicle offloading strategy and server pricing strategy are studied by establishing the game model of the interaction among the edge server, the cloud center and vehicles. The main contributions of this paper are as follows:

- (1) We establish a game model among the edge server, the cloud center and vehicles, and analyze the optimal offloading response of the vehicle based on backward induction. After analysis, we can transform the game problem into a convex optimization problem, and theoretically prove that the game has a unique Nash equilibrium.
- (2) A cloud-edge-vehicle tripartite collaborative task offloading (TCTO) algorithm has been proposed to find the optimal vehicle offloading strategy and server pricing strategy in polynomial time. Numerical experimental results show that, compared with the other algorithms, the proposed algorithm has better performance.

2 Related Works

Due to the limited resources of edge servers, some researches improve the QoS of tasks by expanding the resources of edge networks. In VEC, Sun [7] et al. propose a strategy that can share part of the offloading task by neighboring vehicles, thereby realizing vehicle resource sharing. However, the computing and storage resources of vehicles are often limited, and how to efficiently and stably summon neighboring vehicles remains to be solved. Zhao et al. [8] proposed a cloud-edge collaborative task offloading method, and proposed a distributed algorithm to solve the non-convex problem of offloading decision-making and computing resource allocation. Although the cloud center can provide stable services compared to neighboring vehicles, it ignores the fact that vehicles also have certain task processing capabilities. Pham et al. [9] proposed to use mobile volunteer nodes as an extension of edge server and cloud server resources, which can alleviate the resource constraints of edge networks to a certain extent. However, this scheme does not consider the task offloading decision under the hybrid strategy.

From a market perspective, the vehicle needs to pay a certain amount for the services provided by the server. Therefore, in task offloading, we not only need to consider the optimal offloading strategy of vehicles but also the optimal pricing strategy of server resources. Zeng et al. [10] proposed to use volunteer vehicles to expand the resources of the edge server, and obtain the optimal unloading decision and pricing strategy by constructing a game model of the interaction

between the request vehicles and the edge servers and the volunteer vehicles. However, volunteer vehicles are often difficult to call. Similarly, Yang [11] and Liu [12] also analyzed the optimal task offloading strategy and pricing strategy by constructing a game model of the interaction between the request vehicle and the edge server, and theoretically proved that the game has a unique equilibrium. However, the above research only addresses the optimal vehicle pure offloading strategy and server resource pricing strategy. In fact, a hybrid offload strategy is more appropriate for some decomposable offloading tasks.

In addition, the fiber-wireless technology introduced by Zhang [13] and Guo [14] is able to enhance the communication between vehicles and the edge servers and the cloud center, which lays a foundation for the realization of multi-party collaborative task offloading. To fill the gaps in the above studies, in this paper, we consider a hybrid strategy for vehicle offloading, that is, the offloading task of the vehicle can be simultaneously performed locally, be performed on the edge server, and be performed on the cloud center. By constructing a tripartite game model among vehicles, the edge server and the cloud center, the optimal hybrid offloading strategy of vehicle and the optimal resource pricing strategy of the edge server and the cloud center are obtained based on backward induction method.

3 System Model

3.1 Network Model

In Fig. 1, each RSU integrates a VEC server $V = \{v_1, v_2, \dots, v_m\}$. It is supposed that there are n vehicles $R = \{r_1, r_2, \dots, r_n\}$ within the communication range of v_j . Meanwhile, a requesting task can be abstracted as $\{P_i, \rho, \omega_{1,i}, \omega_{2,i}\} (i \in \{1, 2, \dots, n\})$.

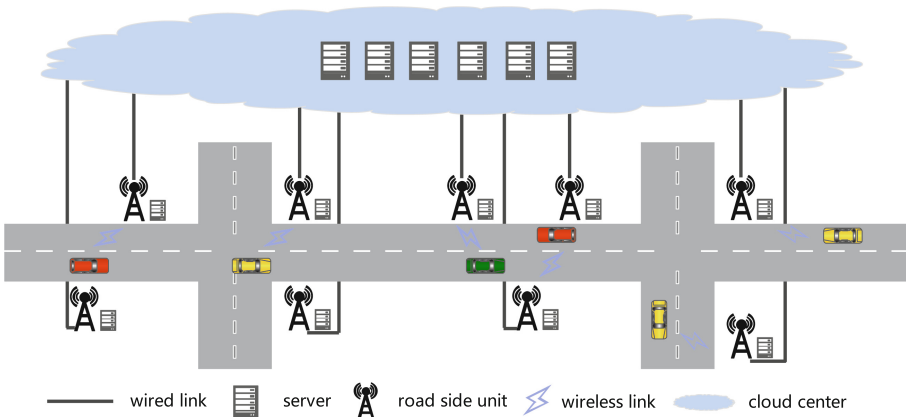


Fig. 1. Network model diagram.

Table 1. Symbol description.

Symbol	Description
P_i/ρ	The task data volume of r_i /unit data computational complexity
$\omega_{1,i}(\omega_{2,i})$	The task proportion of r_i being offloaded to the edge (cloud)
$W_{i,j}(W_{i,c})$	The channel bandwidth between r_i and v_j (cloud)
$h_{i,j}^2(h_{i,c}^2)$	The channel gain between r_i and v_j (cloud)
$\sigma_{i,j}^2(\sigma_{i,c}^2)$	The white gaussian noise between r_i and v_j (cloud)
$R_{i,j}(R_{i,c})$	The data upload rate between r_i and v_j (cloud)
p_i/t_i	The transmission power of r_i /the task offloading delay of r_i
$f_j(f_c, f_i)$	The computing power of the edge (cloud, vehicle)
$\alpha(\beta, \gamma)$	The satisfaction coefficient of the edge (cloud, vehicle)
$\tau_1(\tau_2)$	The proportional gain coefficient of offloading task to the edge (cloud)
$S_{1,j}(S_{2,j})$	The unit resource price for the edge (cloud)
$\vartheta_1(\vartheta_2)$	The delay coefficient of edge (cloud)
$\delta_1(\delta_2, c_i)$	The unit resource cost of the edge (cloud, vehicle)
$\omega_{1,i}^*(\omega_{2,i}^*)$	The optimal task proportion of offloading task to the edge (cloud)
χ_j/t_{max}^i	The resource upper limit of v_j /the task delay limit of r_i

3.2 Communication Model

The communication uses time division multiple access, and we assume that the communication between different vehicles is independent. The data upload rate $R_{i,j}$ and $R_{i,c}$ can be expressed based on the Shannon formula [15].

$$R_{i,j} = W_{i,j} \log_2 \left(1 + \frac{p_i h_{i,j}^2}{\sigma_{i,j}^2} \right), \quad (1)$$

$$R_{i,c} = W_{i,c} \log_2 \left(1 + \frac{p_i h_{i,c}^2}{\sigma_{i,c}^2} \right), \quad (2)$$

Since the data volume of the calculation result is often small, the result receiving time can be ignored. Requesting vehicles also needs data upload time to offload task to edge servers or cloud servers. The task offloading delay t_i can be expressed as:

$$t_i = \max \left\{ \frac{\omega_{1,i} P_i}{R_{i,j}} + \frac{\rho \omega_{1,i} P_i}{f_j}, \frac{\omega_{2,i} P_i}{R_{i,c}} + \frac{\rho \omega_{2,i} P_i}{f_c}, \frac{\rho(1 - \omega_{1,i} - \omega_{2,i}) P_i}{f_l} \right\}, \quad (3)$$

3.3 Utility Functions of Vehicle and Cloud-Edge Alliance

Considering the cost of the vehicle and the quality of the service obtained, the utility function U_i of r_i can be expressed as [16, 17]:

$$\begin{aligned} U_i(\omega_{1,i}, \omega_{2,i}) &= \alpha \ln(1 + \tau_1 \omega_{1,i}) P_i + \beta \ln(1 + \tau_2 \omega_{2,i}) P_i \\ &+ \gamma(1 - \omega_{1,i} - \omega_{2,i}) P_i - \omega_{1,i} \vartheta_1 P_i S_{1,j} - \omega_{2,i} \vartheta_2 P_i S_{2,j} \\ &- (1 - \omega_{1,i} - \omega_{2,i}) c_i P_i. \end{aligned} \quad (4)$$

We maximize the utility of cloud-edge alliance to reflect the cooperative relationship between edge and cloud. Considering the cost and pricing of the edge server and the cloud center, the utility function of cloud-edge alliance U_j can be expressed as:

$$U_j(S_{1,j}, S_{2,j}) = \sum_i \omega_{1,i}(S_{1,j} - \delta_1)P_i + \sum_i \omega_{2,i}(S_{2,j} - \delta_2)P_i, \quad (5)$$

4 Game Theory Based Analysis for Task Offloading

Based on game theory, the task offloading problem proposed in this paper can be decomposed into two maximization problems Q1 and Q2. One is to maximize the utility U_i of the vehicle and the other is to maximize the utility U_j of the edge-cloud alliance.

$$\begin{aligned}
 Q1 : & \max_{\omega_{1,i}, \omega_{2,i}} U_i(\omega_{1,i}^*, \omega_{2,i}^* | S_{1,j}, S_{2,j}) \\
 & s.t. \\
 C1 : & \omega_{1,i} + \omega_{2,i} \leq 1 \\
 C2 : & \omega_{1,i} \geq 0, \omega_{2,i} \geq 0 \\
 C3 : & t_i \leq t_{max}^i
 \end{aligned} \quad (6)$$

$$\begin{aligned}
 Q2 : & \max_{S_{1,j}, S_{2,j}} U_j(S_{1,j}^*, S_{2,j}^* | \omega_{1,i}^*, \omega_{2,i}^*) \\
 & s.t. \\
 C1 : & \sum_i \omega_{1,i} P_i \leq \chi_j \\
 C2 : & S_{1,j} - \delta_1 > 0, S_{2,j} - \delta_2 > 0
 \end{aligned} \quad (7)$$

Lemma 1. *In the game, when $S_{1,j}$ and $S_{2,j}$ are given, the optimal offloading decision of the vehicle r_i is:*

$$\begin{aligned}
 \omega_{1,i}^* &= \frac{\alpha}{\gamma + \vartheta_1 S_{1,j} - c_i} - \frac{1}{\tau_1} \\
 \omega_{2,i}^* &= \frac{\beta}{\gamma + \vartheta_2 S_{2,j} - c_i} - \frac{1}{\tau_2}
 \end{aligned} \quad (8)$$

Proof. When $S_{1,j}$ and $S_{2,j}$ are given, U_i is a function of $\omega_{1,i}$ and $\omega_{2,i}$. the second-order Hessian matrix of U_i can be obtained by calculating the second-order partial derivatives and second-order mixed partial derivatives of $\omega_{1,i}$ and $\omega_{2,i}$ respectively for U_i .

$$\begin{bmatrix} -\frac{\alpha P_i \tau_1^2}{(1 + \tau_1 \omega_{1,i})^2} & 0 \\ 0 & -\frac{\beta P_i \tau_2^2}{(1 + \tau_2 \omega_{2,i})^2} \end{bmatrix}$$

From the second-order Hessian matrix of U_i , we know that U_i is a convex function with respect to $\omega_{1,i}$ and $\omega_{2,i}$ [18]. Therefore, the optimal response (8) of the requesting vehicle can be obtained by setting the first derivative to zero.

Algorithm 1. Tripartite Collaborative Task Offloading Algorithm.

Require: Requesting vehicle tasks volume set $P = \{P_1, P_2, \dots, P_m\}$, utility function U_j , linear constraints @lcon and nonlinear constraints @nonlcon. Set population pop, crossover probability cp, mutation probability mp, maximum evolutionary generation Maxite.

Ensure: The optimal strategy of cloud-edge alliance $S_{1,j}^*$, $S_{2,j}^*$, the best utility U_p .

- 1: initialize pop according to @lcon and @nonlcon
 - 2: **for** ite < Maxite **do**
 - 3: Rank \leftarrow calculate fitness(pop, U_j)
 - 4: pop \leftarrow select population object(pop)
 - 5: pop \leftarrow cross population object(pop, cp)
 - 6: pop \leftarrow mutations(pop, mp)
 - 7: pop \leftarrow remove object(pop, @lcon, @nonlcon)
 - 8: ite \leftarrow ite + 1
 - 9: **end for**
 - 10: return $S_{1,j}^*$, $S_{2,j}^*$, U_p
-

Lemma 2. *The game proposed in this paper has a unique Nash equilibrium.*

Proof. We substitute the optimal offloading response of the vehicle into U_j , and then calculate the second-order partial derivatives and second-order mixed reciprocals of U_j with respect to $S_{1,j}$ and $S_{2,j}$, respectively, to obtain the second-order Hessian matrix with respect to U_j .

$$\begin{bmatrix} -\sum_i \frac{\alpha P_i (\gamma + \vartheta_1 \delta_1 - c_i)}{(\gamma + \vartheta_1 S_{1,j} - c_i)^3} & 0 \\ 0 & -\sum_i \frac{\beta P_i (\gamma + \vartheta_2 \delta_2 - c_i)}{(\gamma + \vartheta_2 S_{2,j} - c_i)^3} \end{bmatrix}$$

where $\gamma + \vartheta_1 \delta_1 > c_i$. From the second-order Hessian matrix of U_j , we know that U_j is a convex function with respect to $S_{1,j}$ and $S_{2,j}$ [18]. Therefore, there are unique $S_{1,j}$ and $S_{2,j}$ for U_j that maximize the utility. In addition, when the maximum utility is achieved, the response of the corresponding vehicle is also optimal, and it is in Nash equilibrium at this time. That is, The game proposed in this paper has a unique Nash equilibrium. Since the solution of game equilibrium is nonlinear, this paper proposed an algorithm 1 that can converge to Nash equilibrium in polynomial time.

5 Simulations

In this section, we will analyze the performance of the TCTO algorithm proposed in this paper and compare it with other algorithms through experiments, such as random algorithm. All experiments were done using Matlab, and the specific parameter values can be found in Table 2.

In Fig. 2(a), when the unit resource pricing is stable, the utility of the corresponding edge server and cloud center reaches the maximum value, which are

0.8899 and 5.3380, respectively. On the contrary, when the unit resource pricing is not stable, the utility of the corresponding edge server and cloud center are 0.7077 and 5.3240 respectively. This result further shows that the TCTO algorithm proposed in this paper can converge to the equilibrium point of the game.

Table 2. Simulation parameters.

Parameter	Value	Parameter	Value
ρ	240	δ_1	0.4
$h_{i,c}/h_{i,j}$	53	δ_2	0.2
$\sigma_{i,c}^2/\sigma_{i,j}^2$	60	τ_1/τ_2	1.0
$W_{i,c}$	5	$W_{i,j}$	10
p_i	0.5	f_j	5
f_c	5	f_l	0.5

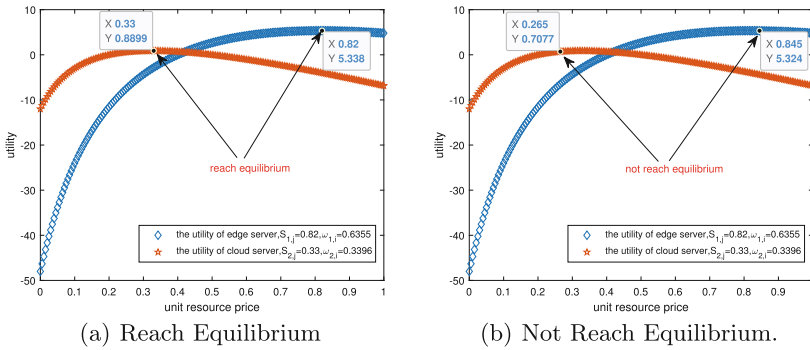


Fig. 2. Utility and equilibrium.

In Fig. 3, diverse execution capabilities vehicles with different task volume will make distinct decisions under the same task offloading scene. Such as, low processing power vehicles with large task volume will offload more task volume to edge servers and cloud servers. In general, the requesting vehicles will select the appropriate task offloading ratio of cloud-edge collaboration according to its processing power and task requirements.

Figure 4 shows the comparison of the performance of the proposed algorithm and random algorithm. It is not difficult to find that the task offloading delay of each vehicle in the TCTO algorithm is significantly lower than the task delay upper limit of 2.5 s. However, under random algorithm, all vehicles other than

car2 did not complete the offloading task within the limited time delay. In addition, the TCTO algorithm also costs less for most of the requested vehicles than the random algorithm. This is because the proposed algorithm is a cloud-edge-vehicle tripartite collaborative task offloading model, which can make full use of the resources of the cloud, the edge and the vehicle to undertake more vehicles offloading tasks under the specified delay requirement.

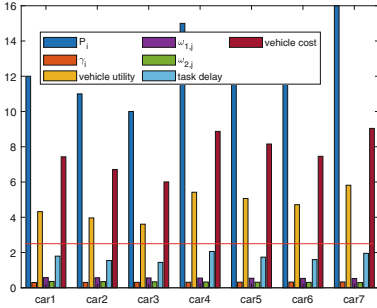


Fig. 3. Vehicles with different capabilities and missions.

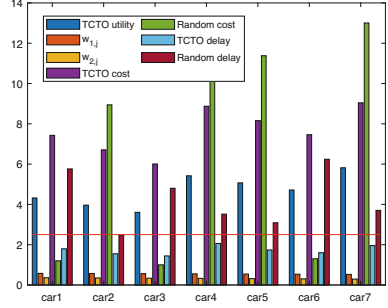


Fig. 4. The performance comparison between TCTO algorithm and random algorithm.

6 Conclusion

In this paper, we consider the hybrid offloading strategy of vehicles and the pricing strategy of edge servers and cloud centers. The optimal offloading strategy and pricing strategy are analyzed by building a game model of vehicles, edge servers and cloud centers. And an algorithm that can find the game equilibrium point in polynomial time is proposed. The numerical experiments show that the TCTO algorithm proposed in this paper has higher performance in terms of delay and cost than the existing algorithms.

Acknowledgements. This work is supported in part by the National Science Foundation of China (Grant No. 62172450), the Key R&D Plan of Hunan Province (Grant No. 2022GK2008) and the Nature Science Foundation of Hunan Province (Grant No. 2020JJ4756). The authors would like to thank the anonymous reviewers for their constructive comments.

References

1. Kekki, S., et al.: MEC in 5G networks. ETSI white paper **28**, 1–28 (2018)
2. Raza, S., Wang, S., Ahmed, M., Anwar, M.R.: A survey on vehicular edge computing: architecture, applications, technical issues, and future directions. *Wirel. Commun. Mob. Comput.* **2019**(2), 1–19 (2019)

3. Liu, L., Chen, C., Pei, Q., Maharjan, S., Zhang, Y.: Vehicular edge computing and networking: a survey. *Mob. Netw. Appl.* **26**(3), 1145–1168 (2021)
4. Liu, J., Wang, S., Wang, J., Liu, C., Yan, Y.: A task oriented computation offloading algorithm for intelligent vehicle network with mobile edge computing. *IEEE Access* **7**, 180491–180502 (2019). <https://doi.org/10.1109/ACCESS.2019.2958883>
5. Zhang, K., Zhu, Y., Leng, S., He, Y., Maharjan, S., Zhang, Y.: Deep learning empowered task offloading for mobile edge computing in urban informatics. *IEEE Internet Things J.* **6**(5), 7635–7647 (2019). <https://doi.org/10.1109/JIOT.2019.2903191>
6. He, X., Lu, H., Du, M., Mao, Y., Wang, K.: QoE-based task offloading with deep reinforcement learning in edge-enabled internet of vehicles. *IEEE Trans. Intell. Transp. Syst.* **22**(4), 2252–2261 (2021). <https://doi.org/10.1109/TITS.2020.3016002>
7. Sun, Y., Guo, X., Zhou, S., Jiang, Z., Liu, X., Niu, Z.: Learning-based task offloading for vehicular cloud computing systems. In: *IEEE International Conference on Communications (ICC)*, pp. 1–7 (2018). <https://doi.org/10.1109/ICC.2018.8422661>
8. Zhao, J., Li, Q., Gong, Y., Zhang, K.: Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks. *IEEE Trans. Veh. Technol.* **68**(8), 7944–7956 (2019). <https://doi.org/10.1109/TVT.2019.2917890>
9. Pham, X.-Q., Nguyen, T.-D., Nguyen, V., Huh, E.-N.: Joint node selection and resource allocation for task offloading in scalable vehicle-assisted multi-access edge computing. *Symmetry* **11**, 58 (2019). <https://doi.org/10.3390/sym11010058>
10. Zeng, F., Chen, Q., Meng, L., Wu, J.: Volunteer assisted collaborative offloading and resource allocation in vehicular edge computing. *IEEE Trans. Intell. Transp. Syst.* **22**(6), 3247–3257 (2021). <https://doi.org/10.1109/TITS.2020.2980422>
11. Yang, F., Yan, J., Guo, Y., Luo, X.: Stackelberg-game-based mechanism for opportunistic data offloading using moving vehicles. *IEEE Access* **7**, 166435–166450 (2019). <https://doi.org/10.1109/ACCESS.2019.2952664>
12. Liu, Y., Wang, S., Huang, J., Yang, F.: A computation offloading algorithm based on game theory for vehicular edge networks. In: *IEEE International Conference on Communications (ICC)*, pp. 1–6 (2018). <https://doi.org/10.1109/ICC.2018.8422240>
13. Zhang, J., Guo, H., Liu, J., Zhang, Y.: Task offloading in vehicular edge computing networks: a load-balancing solution. *IEEE Trans. Veh. Technol.* **69**(2), 2092–2104 (2020). <https://doi.org/10.1109/TVT.2019.2959410>
14. Guo, H., Zhang, J., Liu, J.: FiWi-enhanced vehicular edge computing networks: collaborative task offloading. *IEEE Veh. Technol. Mag.* **14**(1), 45–53 (2019). <https://doi.org/10.1109/MVT.2018.2879537>
15. Shannon, C. E.: A mathematical theory of communication. *Bell Syst. Tech. J.* **27**(3), 379–423 (1948)
16. Weber, E.H.: *Tastsinn und gemeingefühl* (No. 149). W. Engelmann, Leipzig (1905)
17. Fechner, G.T.: *Elemente der psychophysik*, Vol. 2. Breitkopf u. Härtel, Leipzig (1860)
18. Chen, Z.H.: The discrimination method of the concavity and convexity of the binary function and the discussion on the optimum value. *Normal Univ. Sci. J.* **30**(05), 25–28 (2010)



Aerial-Aerial-Ground Computation Offloading Using High Altitude Aerial Vehicle and Mini-drones

Esmail Almosharea¹, Mingchu Li¹(✉), Runfa Zhang¹, Mohammed Albishari¹,
Ikhlal Al-Hammadi¹, Gehad Abdullah Amran¹, and Ebraheem Farea²

¹ Dalian University of Technology, Dalian 116620, China
Esmail20@mail.dlut.edu.cn, mingchul@dlut.edu.cn

² Northeastern University, Shenyang 110169, China

Abstract. Unmanned Aerial Vehicles (UAV) supported by 5G networks can play an important role in providing aerial-aerial/aerial-ground computing services to remote and isolated areas at a low cost. In this paper, we present an aerial-aerial-ground network (AAGN) computing architecture using High Altitude Unmanned Aerial Vehicle (HAU) and Mini-Drones (MDs) based on Mobile Edge Computing (MEC) services where HAU provides computation offloading services for MDs, while MDs can serve as edge computing servers that can be equipped with appropriate capabilities to provide computing services for User Equipments (UEs) on demand. This study focuses on the computation offloading services provided by HAU to MDs, where the MD offloads all or a part of the task to the HAU, and the remaining of the task can be executed by MD. The proposed AAGN framework aims to reduce the MDs' energy consumption and minimize the processing delay by optimizing HAU mobility, MDs scheduling, flight speed, flight angle, and tasks offloading, equipping HAU with the required computing resources. We investigate the computation offloading problem using Deep Deterministic Policy Gradient (DDPG) as a computing offloading approach to learn the optimal offloading policy from a dynamic AAGN environment, considering this problem as a non-convex problem. The simulation results show the feasibility and effectiveness of the proposed AAGN environment where DDPG algorithm can achieve an optimal decision offloading policy and obtains a critical optimization in delay and task offloading ratio compared with Deep Q Network (DQN) and Actor-Critic (AC) algorithms.

Keywords: Aerial computing · High altitude unmanned aerial vehicles · Mobile edge computing · Computation offloading · Reinforcement learning

1 Introduction

The importance of UAVs has been proven to be effective not only on the battlefields but also in other domains, such as natural disasters and providing computing and communication services for remote and isolated areas in a short time and at a low cost [1]. Currently, small and MD are becoming more critical in military and civilian fields where

the missions of the MDs vary according to the purpose for which they are designed, e.g., Monitoring the target area, collecting information and sending them to management and control center, and providing communication and computing services for UEs. But limitations in battery capacity, short flying time, control capabilities, computing resources, and limited area that can be covered represent the most critical challenges for MDs. Enabling aerial-aerial computing and management services for MDs can minimize task latency and reduce battery usage; furthermore, employing a swarm of MDs to perform a mission and the possibility to switch between them can help cover a larger area for a more extended period. Consequently, we propose using AAGN computing architecture where HAU can be used to provide centralized computation offloading services for swarm of MDs, carry out the required processing operations, and offer computing offloading services on-demand. Furthermore, deploying the agile MEC solutions to provide UEs with communication and computing services by MDs is possible. Figure 1 shows our proposed AAGN computing architecture.

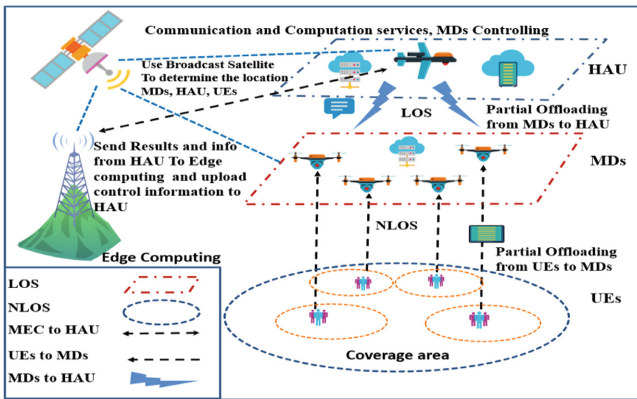


Fig. 1. Our proposed AAGN architecture.

HAU-based aerial computing provides communication and computing offloading services to MDs serving one MD at a time and connected to the edge computing server on the ground, where the MDs perform the tasks assigned to them and offload all or part of the task to HAU on-demand while the rest of the task can be executed by MD. MDs can also serve as an edge computing servers by equipping them with the necessary equipment to provide communication and computation services for UEs. In this paper, we propose a computation and communication dynamic AAGN architecture to provide aerial-aerial/aerial-ground computing services, using DDPG as a computation offloading approach to learn the optimal offloading policy. We can summarize the significant contributions of this study as follows.

- We propose an aerial-aerial/aerial ground computing architecture aiming to deploy dynamic communication and computation offloading services for a swarm of MDs using HAU. We investigate the feasibility of the proposed AAGN environment by

using DDPG to solve computation offloading problems, comparing its computation performance with DQN and AC [2, 3].

- To minimize the processing delay and energy consumption, we optimize HAU mobility, MDs scheduling, flight speed, flight angle, and tasks offloading using DDPG, which supports continuous action space to provide the optimal offloading policy in a proposed AAGN. We emulate this problem as non-convex optimization offloading problems using a Markov Decision Process (MDP) to formulate computing offloading problems, considering the varying in channel status during the flying time of HAU.
- We implemented Simulation within Tensorflow, and evaluated and demonstrated the effectiveness of the proposed AAGN, showing the computation offloading capability of the DDPG algorithm to achieve the best performance (e.g., computation offloading ratio and delay).

The rest of the paper will be covered as follows. Section 2 presents related works. The system model is presented in Sect. 3. The simulation results and discussions are shown in Sect. 4. Finally, we present the conclusion of this study in Sect. 5.

2 Literature Review

There are many recent works related to low and high-altitude computing where UAV-based MEC provides MEC services and internet access for UEs. As for the low-altitude computing, we can present some recent studies. For example, in [4], The authors give an aerial-ground integrated computing system, including a cloudlet server and several MEC servers placed on UAVs to provide UEs with efficient and reliable edge computing services. In [5], UEs obtain computation offloading resources from UAVs to process latency-sensitive tasks. A distributed deep reinforcement learning method is proposed to address the lack of a flexible learning. As for the high-altitude computing, there are some relevant recent studies. For example, in [6], the authors suggest using high altitude platforms (HAPs) and UAVs as part of a hierarchical aerial computing infrastructure to deliver MEC services for terrestrial IoT devices. To dynamically optimize the time and energy consumption for UEs [7], the authors propose a novel framework using high-altitude balloons to provide MEC services for UEs. In [8], the authors proposed a satellite-aerial integrated computing architecture, where UEs can execute their computation tasks locally or offload them to Low Orbit satellite. Previous studies present some frameworks using satellite, HAPs and UAVs to provide aerial-ground computing services for UEs and IoT devices and reduce delays and energy consumption, whereas, in our study, we present a novel aerial-aerial computing framework where an high-altitude UAV provides centralized computing services to MDs, reducing energy consumption and delays for MDs, aiming from this study to reconsider a new architecture for an aerial computing system that is easy to deploy and can cover a larger area on-demand with low cost, keeping in line with the evaluation in MEC, UAV, 5G, and beyond 5G.

3 System Model

3.1 Communication Model

The main purpose of the proposed AAGN is to deploy several MDs in the air where HAU provides them communication and computing services, considering the varying in channel status over time. The communication time T is split out into i time slots. We suppose that MDs keep hovering at a fixed altitude H in a 2D Cartesian coordination system when offloading tasks to HAU. HAU moves in the fixed trajectory manner to cover the MDs target area, where the start coordination of HAU $A(i) = [x(i), y(i)]^T \in \mathbf{R}^{2 \times 1}$ and the final coordination $A(i+1) = [x(i+1), y(i+1)]^T \in \mathbf{R}^{2 \times 1}$ at time slot $i \in \{1, 2, \dots, I\}$. The coordination m of MDs $m \in \{1, 2, \dots, M\}$ is $\mathcal{P}_m(i) = [x_m(i), y_m(i)]^T \in \mathbf{R}^{2 \times 1}$ [9]. The line-of-sight channel gain between the HAU and the MDs can be represented as:

$$g_m(i) = \alpha_0 d_m^{-2}(i) = \frac{\alpha_0}{\|A(i+1) - \mathcal{P}_m(i)\|^2 + H^2}. \quad (1)$$

where $d_m(i)$ denotes the Euclidean distance between HAU and MDs, α_0 indicates the channel gain at a distance $d = 1$ m, and H^2 is HAU fixed flying altitude. The wireless transmission ratio between HAU and MDs can be defined as [10]:

$$r_m(i) = B \log_2 \left(1 + \frac{P_{up} g_m(i)}{\sigma^2} \right), \quad (2)$$

where B represents the network bandwidth, P_{up} represents the upload link transmit power and σ^2 is the noise power.

3.2 Computation Model

A partial offloading strategy is applied in each time slot in the proposed AAGN system, whether at the MDs level or UEs level [16]. Let $\mathcal{R}_{m(i)} \in [0, 1]$ indicates the task offloading ratio between MDs and HAU, and $(1 - \mathcal{R}_{m(i)})$ indicate the remaining tasks that are executed locally by MDs. The delay of MD execution can be denoted as:

$$\mathcal{T}_{MD,m}(i) = \frac{(1 - \mathcal{R}_{m(i)}) \mathcal{C}_m(i), \mathcal{S}}{\mathcal{F}_{MD}}. \quad (3)$$

where $\mathcal{C}_m(i)$ indicates MD's computing task sizes, \mathcal{S} denotes the required CPU cycles to process each unit byte, and \mathcal{F}_{MD} indicates the computing capability of MDs. The HAU starts moving at a time slot i from its initial position $A(i)$ and has to reach the destination $A(i+1) = [x(i) + v(i)\mathcal{T}_{fly} \cos \beta(i), y(i) + v(i)\mathcal{T}_{fly} \sin \beta(i)]^T$ at a speed $v(i) \in [0, v_{max}]$ and an angle $\beta(i) \in [0, 2\pi]$. The flight's energy consumption is calculated as follows:

$$\mathcal{E}_{fly}(i) = \varnothing \|v(i)\|^2. \quad (4)$$

where $\varnothing = 0.5W_{HAU}\mathcal{T}_{fly}$, W is the payload of the HAU, and the flight time \mathcal{T}_{fly} is a fixed value [4]. We can denote the transmission delay as follows:

$$\mathcal{T}_{tra,m}(i) = \frac{\mathcal{R}_{m(i)} \mathcal{C}_m(i) \mathcal{S}}{r_m(i)}. \quad (5)$$

where the computation delay at HAU server can be represented as:

$$\mathcal{T}_{HAU,m}(i) = \frac{\mathcal{R}_m(i)\mathcal{C}_m(i)\mathcal{S}}{\mathcal{F}_{HAU}}. \quad (6)$$

\mathcal{F}_{HAU} represents HAU server CPU's computation frequency. When the computation process is performed on HAU server, the energy consumption can be represented as:

$$\mathcal{P}_{HAU,m}(i) = M\mathcal{F}_{HAU}^3, \quad (7)$$

Then, the energy consumption of HAU server at time slot i can be represented as:

$$\mathcal{E}_{HAU,m}(i) = \mathcal{P}_{HAU,m}(i)\mathcal{T}_{HAU,m}(i) = M\mathcal{F}_{HAU}^2\mathcal{R}_m(i)\mathcal{C}_m(i)\mathcal{S}, \quad (8)$$

3.3 Problem Formulation

We optimize HAU mobility, resource management, MDs scheduling, and task offloading to ensure the best utilization of computation resources and to reduce processing delays benefiting from the line of sight between HAU and MDs. We can present the optimization problems as follow:

$$\min_{\{u_m(i), \mathbf{A}(i+1), \mathcal{R}_m(i)\}} \sum_{i=1}^I \sum_{m=1}^M u_m(i) \max\{\mathcal{T}_{MD,m}(i), \mathcal{T}_{HAU,m}(i) + \mathcal{T}_{tra,m}(i)\} \quad (C1)$$

$$u_m(i) \in \{0, 1\}, \forall i \in \{1, 2, \dots, I\}, m \in \{1, 2, \dots, M\} \quad (C2)$$

$$\sum_{m=1}^M u_m(i) = 1, \forall i \quad (C3)$$

$$0 \leq \mathcal{R}_m(i) \leq 1, \forall i, m \quad (C4)$$

$$\mathbf{A}(i) \in \{(x(i), y(i)) | x(i) \in [0, L], y \in [0, W]\}, \forall i \quad (C5)$$

$$\mathcal{P}(i) \in \{(x_m(i), y_m(i)) | x_m(i) \in [0, L], y_m \in [0, W]\}, \forall i, m \quad (C6)$$

$$\sum_{i=1}^I (\mathcal{E}_{fly,m}(i) + \mathcal{E}_{HAU,m}(i)) \leq \mathcal{E}_b, \forall m \quad (C7)$$

$$\sum_{i=1}^I \sum_{m=1}^M u_m(i)\mathcal{C}_m(i) = \mathcal{C} \quad (C8)$$

C2 and C3 ensure that in time slot i , only one MD can be scheduled for computation. C4 represents the value range for the computing task's offloading ratio. C5 and C6 denote that HAU and MDs fly inside the defined area. C7 guarantees that the total energy consumed by HAU during flying and computation doesn't overpass the battery energy. C8 states that all computation tasks must be implemented within the ensured period.

4 The Simulation Results

This section clarifies the scenarios and simulation parameters in detail to investigate the computation offloading in AAGN environment and evaluate the DDPG algorithm's performance compared with DQN and AC algorithms.

4.1 Simulation Setting

In the proposed AAGN system, we consider 4 MDs and one HAU distributed over a 2 Dimensional Coordinate area $L \times W = 3000 \times 3000 \text{ m}^2$. MDs hover at 300 m altitude. According to the scenario, HAV gross weight $W_{HAU} = 150 \text{ kg}$ [11], flies at a fixed altitude $H = 2500 \text{ m}$. The time period $T = 400 \text{ s}$ is split into $I = 40 \text{ s}$ time slots, in each time slot HAU flight time is $T_{fly} = 1 \text{ s}$. The maximum HAU and MD flight speed are $V_{max} = 50 \text{ m/s}$. We set the channel power gain to be $a_0 = -50 \text{ dB}$ with $d = 1 \text{ m}$ as a reference distance [19], and the transmission bandwidth for MDs $B_{MD} = 2 \text{ MHz}$ [12]. The receiver noise power is set to be $\sigma^2 = -100 \text{ dBm}$ without signal blockage [13]. We considered that the MD's transmission power $\mathcal{P}_{UPMD} = 0.8 \text{ W}$. The capacity of the HAU battery is $\mathcal{E}_B = 1000 \text{ KJ}$. The frequency cycles of CPU for HAU and MDs required per bit are $= 3000$ and $= 1000$ cycles/bit, respectively [14]. MDs and HAU computing capability is set to $\mathcal{F}_{MD} = 2.4 \text{ GHz}$, and $\mathcal{F}_{HAU} = 5.8 \text{ GHz}$, respectively. We evaluate the computation offloading performance for DDPG algorithm compared with DQN algorithm and AC and investigate the feasibility of the proposed AAGN framework.

4.2 Parametric Analyses

To evaluate DDPG performance as a computation offloading solution for the proposed AAGN environment, we perform several experiments to determine the optimal values for a_{actor} learning rate, a_{critic} learning rate, discount factor γ and exploration variable σ_e . We can deduce that the DDPG algorithm converges when $a_{actor} = 0.1$, $a_{critic} = 0.2$ and $a_{actor} = 0.001$, $a_{critic} = 0.002$ and cannot converge when $a_{actor} = 0.00001$, $a_{critic} = 0.00002$, because DNN updates are slower when using lower learning rates. So, $a_{actor} = 0.001$ and $a_{critic} = 0.002$ are considered the optimal learning rates. The computation offloading policy in the training stage achieves high performance using the discount factor $\gamma = 0.001$ due to the substantial changes in the environment over time. A larger γ denotes that the data collected during the period will be considered by the Q table as long-term data that results in poor generalization over time. As a result, an acceptable value of γ can enhance the eventually trained policy performance, and in the following experiments, we set the discount factory γ to 0.001. A larger value of the exploration parameter results in a larger random noise in the distribution, allowing the agent to explore a larger spatial range. When $\sigma_e = 0.005$, the algorithm's performance reaches a locally optimal solution at 200 episodes because of the small exploration parameters, so we use it in the experiments.

4.3 Performance Evaluation

We compare DDPG performance with DQN and AC using different task sizes under AAGN environment, evaluating the efficiency of AAGN. In Fig. 2a, when task size $\mathcal{D} = 1 \text{ MB}$, DDPG and AC algorithms can achieve higher performance than the DQN algorithm since both algorithms deal easily with continuous action spaces. In contrast, DQN cannot optimally determine the best offloading strategy because of the search for non-negligible gaps and discrete action space among possible actions. In Fig. 2b, when task size $\mathcal{D} = 15 \text{ MB}$, both DDPG and DQN achieve higher performance than the AC algorithm. That's because, unlike DDPG and DQN, the AC algorithm doesn't use a Reply Buffer to store the learning model, but learns the models at every step. DDPG algorithm keeps low processing delay and high convergence with the more significant task size.

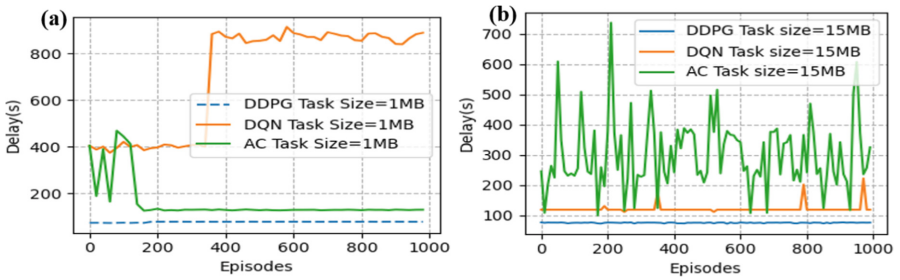


Fig. 2. The performance of the DDPG algorithm compared with DQN and AC using different task sizes.

Figure 3 shows The processing delay performance and offloading ratio when task size $\mathcal{D} = 3.5 \text{ MB}$ and computing capability of MD $\mathcal{F}_{MD} = 2.4 \text{ GHz}$. DDPG optimization schemes can optimize the convergence performance and achieve lower processing delay because the action space is continuous, whereas DQN's performance suffers significantly when attempts to build a state-action value for each action. Discretizing the action space is one of the most effective approaches to solving this challenge, but it cannot obtain the optimal solution. This explains the high performance of the DDPG algorithm's since it deals well with continuous action spaces, allowing the output layer in the action model to be adjusted to fit the environment's action space. In terms of offloading ratio, Fig. 3b shows the offloading ratio where DDPG algorithm achieves an optimal offloading ratio. However, in the AC algorithm, when the computational capabilities are high, MDs tend to perform the tasks locally, which explains why AC processing delay is less than DQN, while DQN can achieve a better offloading ratio than AC.

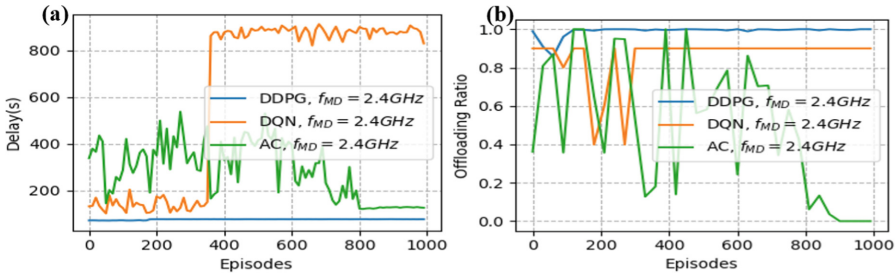


Fig. 3. The processing delay performance and offloading ratio of the DDPG algorithm compared with DQN and AC using task size $D = 3.5$ MB.

However, the DQN algorithm obtains a lower offloading ratio compared with DDPG but still can achieve an offloading convergence. From Fig. 2 and Fig. 3, we notice that when the task size increases, the DQN algorithm achieves more convergences. In all previous Figures, the DDPG algorithm is the optimal computation offloading approach for our proposed AAGN framework. To summarize, the efficiency of the proposed AAGN has been investigated. DDPG algorithms achieve optimal convergence with lower processing delay and a high offloading ratio. Therefore, DDPG can be a suitable solution for computation offloading problems under a dynamic network environment (e.g., AAGN).

5 Conclusion

This paper presents an aerial-aerial/aerial-ground computing architecture using HAU and a swarm of MDs based on MEC services, where the HAU provides communication and computing services to the MDs. The proposed architecture aims to find solutions for the MDs' limited battery capacity, limited coverage area, and short flying time. We study in detail the computation offloading processes between MDs and HAU. The efficiency of the proposed AAGN architecture was estimated by evaluating the processing delay and offloading ratio. To achieve high performance, we optimize HAU mobility, MDs scheduling, flight speed, flight angle, and tasks offloading using DDPG to solve the computation offloading problem, considering this problem as a non-convex problem. The results show the feasibility and effectiveness of our proposed AAGN architecture using the DDPG algorithm as a computation offloading policy which can achieve optimal results in a processing delay and offloading ratio compared with DQN and AC algorithms.

References

1. Gao, A., Wang, Q., Hu, Y., Duan, W.: An offloading optimization scheme for multi-UAV aided network in mobile computing. In: Proceedings of the 2020 International Wireless Communications and Mobile Computing Conference, IWCMC 2020, no. 2018, pp. 1468–1473 (2020). <https://doi.org/10.1109/IWCMC48107.2020.9148136>
2. Munk, J., Kober, J., Babuska, R.: Learning state representation for deep actor-critic control. In: Proceedings of the 2016 IEEE 55th Conference on Decision and Control CDC 2016, pp. 4667–4673 (2016). <https://doi.org/10.1109/CDC.2016.7798980>

3. Ohnishi, S., Uchibe, E., Yamaguchi, Y., Nakanishi, K., Yasui, Y., Ishii, S.: Constrained deep Q-learning gradually approaching ordinary Q-learning. *Front. Neurobot.* **13**, 1–19 (2019). <https://doi.org/10.3389/fnbot.2019.00103>
4. Hu, Q., Cai, Y., Yu, G., Qin, Z., Zhao, M., Li, G.Y.: Joint offloading and trajectory design for UAV-enabled mobile edge computing systems. *IEEE Internet Things J.* **2019**(CP768), 31–37 (2019)
5. Wei, D., Ma, J., Luo, L., Wang, Y., He, L., Li, X.: Computation offloading over multi-UAV MEC network : a distributed deep reinforcement learning approach. *Comput. Netw.* **199**, 108439 (2021). <https://doi.org/10.1016/j.comnet.2021.108439>
6. Jia, Z., Wu, Q., Member, S., Dong, C.: Hierarchical Aerial Computing for Internet of Things via Cooperation of HAPs and UAVs, pp. 1–12
7. Wang, S., et al.: Federated learning for task and resource allocation in wireless high-altitude balloon networks. *IEEE Internet Things J.* **8**(24), 17460–17475 (2021). <https://doi.org/10.1109/JIOT.2021.3080078>
8. Zhang, L., Zhang, H., Guo, C., Xu, H., Song, L., Han, Z.: Satellite-aerial integrated computing in disasters: user association and offloading decision. In: *Proceedings of the IEEE International Conference on Communications*, vol. 2020 (2020). <https://doi.org/10.1109/ICC40277.2020.9148796>
9. Diao, X., Zheng, J., Cai, Y., Wu, Y., Anpalagan, A.: Fair data allocation and trajectory optimization for UAV-assisted mobile edge computing. *IEEE Commun. Lett.* **23**(12), 2357–2361 (2019). <https://doi.org/10.1109/LCOMM.2019.2943461>
10. Coldrey, M., Berg, J.E., Manholm, L., Larsson, C., Hansryd, J.: Non-line-of-sight small cell backhauling using microwave technology. *IEEE Commun. Mag.* **51**(9), 78–84 (2013). <https://doi.org/10.1109/MCOM.2013.6588654>
11. Boukoberine, M.N., Zhou, Z., Benbouzid, M.: A critical review on unmanned aerial vehicles power supply and energy management: solutions, strategies, and prospects. *Appl. Energy* **255**, 113823 (2019). <https://doi.org/10.1016/j.apenergy.2019.113823>
12. Wang, Y., Fang, W., Ding, Y., Xiong, N.: Computation offloading optimization for UAV-assisted mobile edge computing: a deep deterministic policy gradient approach. *Wireless Netw.* **27**(4), 2991–3006 (2021). <https://doi.org/10.1007/s11276-021-02632-z>
13. Jeong, S., Simeone, O., Kang, J.: Mobile edge computing via a UAV-mounted cloudlet: optimization of bit allocation and path planning. *IEEE Trans. Veh. Technol.* **67**(3), 2049–2063 (2018). <https://doi.org/10.1109/TVT.2017.2706308>
14. Xiong, J., Guo, H., Liu, J.: Task offloading in UAV-aided edge computing: bit allocation and trajectory optimization. *IEEE Commun. Lett.* **23**(3), 538–541 (2019). <https://doi.org/10.1109/LCOMM.2019.2891662>



Meta-MADDPG: Achieving Transfer-Enhanced MEC Scheduling via Meta Reinforcement Learning

Yiming Yao^{1,4}, Tao Ren^{1,4}(✉), Meng Cui², Dong Liu³, and Jianwei Niu^{4,1}

¹ Hangzhou Innovation Institute, Beihang University, Hangzhou 310051, China
yymmine@126.com, taotao_1982@126.com

² CNPC Engineering Technology R&D Company Limited, Beijing 102206, China
cuimengdri@cnpc.com.cn

³ College of Computer and Information Engineering, Henan Normal University,
Xinxiang 453007, China

⁴ State Key Laboratory of Virtual Reality Technology and Systems, School of Computer
Science and Engineering, Beihang University, Beijing 100191, China
niu Jianwei@buaa.edu.cn

Abstract. With the assistance of mobile edge computing (MEC), mobile devices (MDs) can optionally offload local computationally heavy tasks to edge servers that are generally deployed at the edge of networks. As thus, the latency of task and energy consumption of MDs can be both reduced, significantly improving mobile users' quality of experience. Although considerable MEC scheduling algorithms have been designed by researchers, most of them are trained to solve specific tasks, leaving the performance in other MEC environments remaining dubious. To address the issue, this paper first formulates the optimization problem to minimize both task delay and energy consumption, and then transforms it into Markov decision problem that is further solved by using the state-of-the-art multi-agent deep reinforcement learning method, i.e., MADDPG. Furthermore, aiming at improving the overall performance in various MEC environments, we integrate MADDPG with meta-learning and propose Meta-MADDPG which is carefully designed with dedicated reward functions. The evaluation results are given to showcase the more satisfactory performances of Meta-MADDPG over the state-of-the-art algorithms when confronting new environments.

Keywords: Deep reinforcement learning · Multi-agent deep deterministic policy gradient · Meta learning · Mobile edge computing

1 Introduction

In recent decade, mobile applications (APPs) have been thriving with the emergence of numerous popular and cutting-edge applications, e.g., face recognition, real-time online games, and virtual reality, which are usually computation-intensive and delay-sensitive. Nonetheless, finite computing capacity and battery power have strained mobile

devices (MDs) from meeting abovementioned requirements of APPs. Motivated by this, mobile edge computing (MEC) is proposed as a promising technology to mitigate the gap between MDs' limited capacity and APPs' increasing requirements [1]. With MEC, MDs can choose to offload their tasks to edge servers, which are generally deployed at the edge of networks and equipped with resilient distributed computing resources and energy supply.

A non-trivial issue in MEC is computation offloading decision making and tasks scheduling [2]. In view of the superiority in optimal control, deep reinforcement learning (DRL) has attracted increasing attention from both academia and industry [3], resulting in various DRL-based MEC algorithms for scheduling computation offloading tasks. The researchers in [4] advanced a novel strategy for jointly optimizing task offloading and resource allocation in dynamic MEC. [5] presented a two-layer optimization algorithm to optimize UAV deployment and task scheduling to achieve minimal system energy consumption. [6] proposed to use multi-agent depth deterministic strategy gradient (MADDPG) to maximize offloading tasks while meeting the requirements of heterogeneous service quality. However, the existing DRL-based MEC algorithms are too specialized to achieve best performances when encountering new MEC environments.

To address the issue, we propose a novel transfer-enhanced computation offloading and resource allocation algorithm based on deep reinforcement learning. We first formulate the optimization problem with minimizing the weighted sum of task delay and energy consumption, and then transform it into Markov decision problem which is further solved by using multi-agent deep deterministic strategy gradient (MADDPG) [8]. Furthermore, we integrate MADDPG with meta-learning and propose the Meta-MADDPG algorithm which is able to adapt its scheduling policy to new MEC environments. Ordinary DRL learns the optimal policy under a specific MDP, while Meta-RL can quickly adapt to different new MDPs and get the corresponding optimal policy [9]. In the Meta-RL scenario, our goal is to learn a learning process that can quickly adapt to a new MDP task with very few samples. The main contributions of this paper are summarized as follows:

- We set the MEC environment divided by the large span change of the data size, required computing power, and delay sensitivity of the tasks. In other words, the data size range, required computing power range, and delay sensitivity of the tasks in each MEC environment are unique.
- We set experience pool for each pair of agent and environment to distill scheduling policies from various MEC environments and sample experiences from these pools according to their contribution to the policy distillation.
- We design a novel algorithm of fast adaptive capacity, named Meta-MADDPG, to re-sample MADDPG network after each update. In addition, we perform periodic parameter initialization on the network every longitudinal time for all agents, so as to find promising initial parameters suitable for various environments.
- We compare the performance of the proposed Meta-MADDPG with the state-of-the-art algorithms. Simulation results show that Meta-MADDPG could achieve not only faster convergence but also lower task delay and energy consumption in different MEC environments.

The rest of this paper is organized as follows. In Sect. 2, we present the system model, followed by the formulation of the optimization problem in Sect. 3. Section 4 introduces the Meta-MADDPG algorithm, along with a parameter initialization method based on deep reinforcement learning. In Sect. 5 and 6, we discuss the evaluation results and give the conclusion, respectively.

2 System Model

2.1 MEC Network

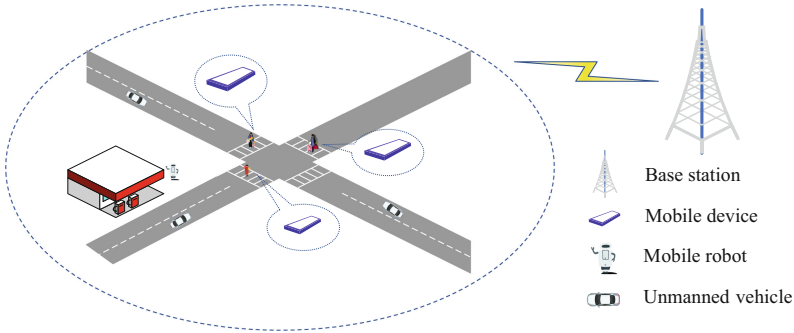


Fig. 1. Multi-agent MEC system model.

Figure 1 shows a MEC network structure consisting of multiple mobile devices and a single base station (BS). The system time is evenly divided into N slots, $\mathcal{N} = \{1, 2, \dots, N\}$. MDs at the MD layer are denoted by the set $\mathcal{M} = \{1, 2, \dots, M\}$. The computing resources and battery resources embedded in each MD are expressed as f^{MD} and e^{MD} , respectively. In addition, the idle power of each MD is defined as p^{idle} . Each mobile device periodically generates computing tasks with different QoS requirements. The device can choose to execute the computing tasks locally or at the edge server.

At the Edge layer, we posit BS as $L = (x^b, y^b, H^b)$. The computing ability of additional ES of BS is f^{ES} , and the computing ability allocated to the task offloaded to ES is $f_n^{m,ES}$. Compared with the wireless transmission delay between MDs and BS, the transmission delay between BS and its ES is negligible because the result of the task calculated by BS is often much smaller than the input task. Tasks will be transferred via a communication link if the devices choose to offload them to the edge servers. If multiple mobile devices offload computing tasks simultaneously, the wireless bandwidth will be evenly allocated to perform the offloading tasks.

2.2 Task Model

In slot n , each mobile device generates the calculation task g_n^m with different QoS requirements. Task g_n^m consists of three elements [10], namely $g_n^m = (D_n^m, Z_n^m, \tau^m)$, where D_n^m, Z_n^m, τ^m represent the input task data (in bits), CPU cycles per bit of task data, and the maximum allowable task delay, respectively. All mobile devices can execute tasks locally or offload tasks to the edge servers. τ^m represents the maximum allowable delay of each computation task, which is a crucial index of QoS. These three parameters are different for each task.

The edge offloading indicator of the m th MD at the n th slot is defined as: $\kappa_n^m \in \{0, 1\}, \forall m \in \mathcal{M}, n \in \mathcal{N}$, where $\kappa_n^m = 0$ indicates local execution, and $\kappa_n^m = 1$ indicates task offloading to the edge server. In this paper, we adopt a binary offload strategy, where the task is entirely executed either locally on the MD or remotely at the edge server. $K = [\kappa_1, \kappa_2, \dots, \kappa_n^m]$ denotes offloading decision variables.

2.3 Computing Model

– Local Computation Model

If the indicator $\kappa_n^m = 0$, MD m chooses to execute the calculation locally, g_n^m will be executed on MD m . Set the transmitting power of the mobile device to 0 and the computing capacity of the base station to 0. The local computation time for MD m is:

$$T_n^{m,1} = \frac{D_n^m \times Z_n^m}{f_n^m}, \quad (1)$$

where f_n^m refers to the computing power allocated to the task by the mobile device, that is, the number of CPU cycles per second.

Meanwhile, we can express the energy consumed by the local execution of any task as:

$$E_n^{m,1} = 10^{-27} \times (f_n^m)^2 \times D_n^m \times Z_n^m, \quad (2)$$

where 10^{-27} represents the chip-dependent calculation coefficient, which is the energy consumed per CPU cycle during the execution of the task. According to one literature [11] we set it to be 10^{-27} .

We set two weight parameters ω_i^t and ω_i^e related to $T_n^{m,1}$ and $E_n^{m,1}$ respectively, and express the weighted sum of local calculation time and energy consumption as C_n^1 .

– Edge Computation Model

If the offloading indicator $\kappa_n^m = 1$, MD m chooses to offload the task to the edge layer, g_n^m must be transmitted through the wireless connection between MD m and BS. Since there may be multiple MD-BS connections in the same slot, signal interference must be considered. The white Gaussian noise power is expressed as σ^2 . The SINR of BS receiving g_n^m can be calculated as follows:

$$SINR_n^m = \frac{\eta_n^m p_n^{\text{MD}}}{\sum_{m' \neq m \& \kappa_{n'}^{m'} \neq 0} \eta_{n'}^{m'} p_n^{\text{MD}} + \sigma^2}, \quad (3)$$

where p^{MD} is the transmitted power of MD. The channel gain η_n^m [12] as follows:

$$\eta_n^m = \theta(\delta_n^m)^{-\alpha} ||h||^2 \zeta. \tag{4}$$

where δ_n^m is the distance between MDs and BS. We can obtain that the wireless transmission rate from MD m to BS is:

$$\gamma_n^m = B \log(1 + \text{SINR}_n^m), \tag{5}$$

where B represents the wireless channel bandwidth of BS. Using γ_n^m , we can get the upload time $T_n^{m,o,u}$ of g_n^m during the offloading phase:

$$T_n^{m,o,u} = \frac{D_n^m}{\gamma_n^m}. \tag{6}$$

In addition, the upload energy consumption of MD m offloading g_n^m can be calculated as:

$$E_n^{m,o,u} = p^{\text{MD}} T_n^{m,o,u} = \frac{p^{\text{MD}} D_n^m}{\gamma_n^m}. \tag{7}$$

Then the calculation time of BS executing g_n^m is as follows:

$$T_n^{m,o,c} = \frac{Z_n^m D_n^m}{f_n^{m,ES}}. \tag{8}$$

Therefore, the total delay of the offloading process can be obtained $T_n^{m,o}$. In the offloading process, the mobile device consumes idle energy inevitably for sustaining operation, which can be expressed as follows:

$$E_n^{\text{idle}} = p^{\text{idle}} T_n^{m,o,c} = \frac{p^{\text{idle}} Z_n^m D_n^m}{f_n^{m,ES}}. \tag{9}$$

The total energy consumption of offloading process is $E_n^{m,o}$. We relate two weight parameters ω_o^i and ω_o^e to $T_n^{m,o}$ and $E_n^{m,o}$, respectively. Therefore, the total cost of the offloading task is C_n^o , the total cost of local computation is C_n^1 , and the total cost of all mobile devices is expressed as follows:

$$C_{all} = \sum_{i=1}^n [(1 - \kappa_n^m) C_n^1 + \kappa_n^m C_n^o]. \tag{10}$$

3 Problem Formulation

This section proposes optimization problems for computational offloading and resource allocation based on MEC systems. Under the following constraints, we aim to minimize the total cost of execution time and energy consumption of all mobile devices in the system. The problem can be formulated as:

$$\begin{aligned}
 \mathcal{P} = & \min_{(f_n^m, \kappa_n^m)} \sum_{i=1}^n [(1 - \kappa_n^m) C_n^1 + \kappa_n^m C_n^o] \\
 \text{s. t. } & C1 \boxtimes \kappa_n^m \in \{0, 1\}, \forall m \in \mathcal{M}, n \in \mathcal{N} \\
 & C2 \boxtimes f_n^m \leq f^{\text{MD}}, \forall m \in \mathcal{M}, n \in \mathcal{N} \\
 & C3: \sum_{m \in \mathcal{M}, n \in \mathcal{N}} \kappa_n^m f_n^{m, \text{ES}} \leq f^{\text{ES}} \\
 & C4 \boxtimes E_n^{m, \text{left}} = E^{\text{MD}} - \sum_{n=1}^N E_n^m \geq 0, \forall m \in \mathcal{M}, n \in \mathcal{N} \\
 & C5 \boxtimes T_n^m \leq \tau^m, \forall m \in \mathcal{M}, n \in \mathcal{N},
 \end{aligned} \tag{11}$$

where T_n^m is the execution delay of each task g_n^m , and E_n^m is the energy consumption of MD m at slot n . In Formula (11), edge offloading decision variables can only be discrete values (C1). The allocated computing resources should not exceed the available computing resources of the MD (C2) (C3). The remaining energy $E_{n'}^{m, \text{left}}$ for each MD m of any time slot n' should be nonnegative (C4). Finally, the delay of g_n^m per task should not exceed the maximum allowable delay (C5).

Formula (11) achieved optimization by finding the optimal offloading decision vector and resource allocation. It can be seen that the objective function C_{all} contains integer κ_n^m and continuous f_n^m mixed optimization variables. In addition, we found that the objective function C_{all} and constraints C2, C3, C4 and C5 are nonlinear. Since K is a binary variable and the objective function is not convex, the optimization problem \mathcal{P} can be translated as a mixed integer nonlinear programming (MINLP) problem. When mobile devices increase, the problem expands substantially, usually NP-hard [13]. Since the interaction between mobile devices and the environment corresponds to the Markov decision process (MDP), we propose solving the NP-hard problem using the reinforcement learning method.

4 Problem Solution

In this section, we introduce reinforcement learning (RL) including three key parameters [14], and introduce a classical RL learning method deep deterministic policy gradient (DDPG). On this basis, we propose Meta-MADDPG, a reinforcement learning method based on meta-learning and DDPG, which can better solve the problems proposed in this paper.

4.1 RL

Three critical parameters in reinforcement learning, state, action, and reward, can define MDP.

- state: The state refers to the collective states of all the involved mobile devices, mainly consisting of dynamic information. In each slot n , the state can be expressed as the elements of $s_n = \{g_n^m, \uparrow_n^m, E_n^{m,\text{left}} | m \in \mathcal{M}, n \in \mathcal{N}\}$. The message of s_n includes task data size, task computing size and task tolerance delay, current energy, channel gain to the base station.
- action: The action represents the set of MDs' actions, the set of scheduling variables. The message of each action includes the offloading decision variables, transmission power, and the mobile device's computing power. In each slot n , a_n can be expressed as $a_n = \{\kappa_n^m, f_n^m | m \in \mathcal{M}, n \in \mathcal{N}\}$.
- reward: It represents the reward value of each mobile device after executing action a in state s . In addition, we set up a reward function r , denoted by $r_a^{s \rightarrow s'}$, which determines how much reward value should be given immediately when the action a is taken in state s transitions to state s' .

4.2 DDPG

DDPG is a typical reinforcement learning method for continuous behavior. Firstly, a deterministic behavior strategy μ is defined, and the action of each step can be calculated by $a_t = \mu(s_t)$. Then we use a neural network to simulate the function μ , called policy network, with parameters θ^μ . In training, random noise is introduced into the decision-making mechanism of action, and the decision of action is changed from a deterministic process to a random process. Then the action is sampled from this process and transmitted to the environment for execution.

After passing action a_t to the environment for execution under a state s_t , an expected value R_t will be obtained, namely the action-value function, which is defined by Bellman equation as:

$$Q^\mu(s_t, a_t) = E[r(s_t, a_t) + \gamma Q^\mu(s_{t+1}, \mu(s_{t+1}))]. \quad (12)$$

As Q is a recursive expression, we use a convolution neural network in DDPG to simulate the Q function with the parameter θ^Q , and use a function to measure the performance of strategy μ , which is defined as:

$$\begin{aligned} J_{\beta(\mu)} &= \int \rho^\beta(s) Q^\mu(s, \mu(s)) ds \\ &= E_{s \sim \rho^\beta} [Q^\mu(s, \mu(s))] \end{aligned} \quad (13)$$

Therefore, the optimal behavior strategy μ is obtained, i.e., maximizes the μ of $J_{\beta(\mu)}$:

$$\mu = \arg \max_{\mu} J(\mu). \tag{14}$$

4.3 Meta-MADDPG

On the basis of defining MDP, we first transform the optimization problem \mathcal{P} into the strategy of pursuing optimal deterministic action decision $\mathbf{a}_n = \boldsymbol{\pi}(s_n), \forall n \in N$, with the goal of maximizing long-term reward as follows:

$$\mathcal{P}' = \arg \max_{\pi} q_{\pi}(s_0, a_0), \tag{15}$$

where $q_{\pi}(s_0, a_0)$ is the maximum long-term return expected by a at s :

$$q_{\pi}(s, \mathbf{a}) = E \left[R_a^{s \rightarrow s'} + \gamma \max_{s'} q_{\pi}(s', a') \right]. \tag{16}$$

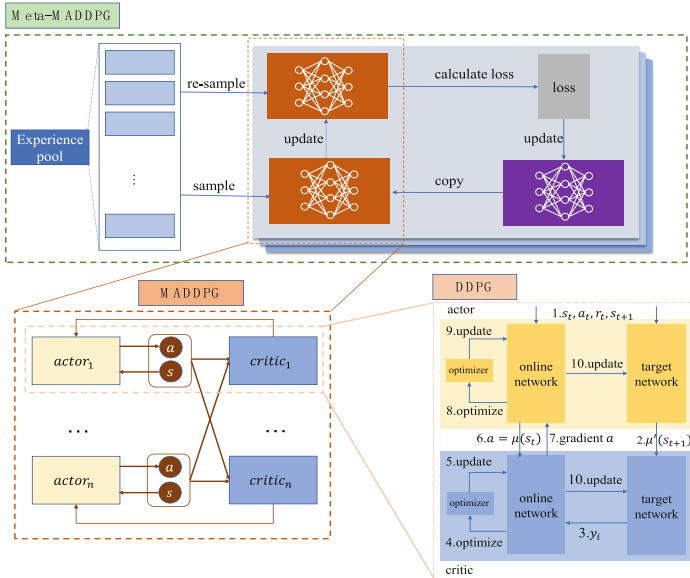


Fig. 2. Diagram of Meta-MADDPG's network structure.

Meta-MADDPG is divided into inner and outer structure, as shown in Fig. 2. At the beginning of training, get state s_n after observing the environment, and generate a random action a_n from one noise variable random sampling. Then execute a_n , get observations s_{n+1} , reward r_n , package (s_n, a_n, r_n, s_{n+1}) and store in experience pool \mathbb{B}_x . After the experience pool \mathbb{B}_x is full, we take S samples.

In the inner structure, we use MADDPG network and take state, action, reward, and next state information of each mobile device from the experience pool into actor network and critic network. Then the soft update is used to update the target network. After that, the information in the experience pool is passed to the updated DDPG network, and the loss value of the updated network is calculated.

In the outer structure, the loss value calculated by the inner structure is used to update the parameters of the meta-policy. Set a value ψ , after each training ψ wheel, we use the average value of loss calculated by the inner structure to update the parameters of the outer structure. Finally, all the parameters of the outer structure are copied and initialized as the parameters of the inner structure. Repeat the cycle. After each training ψ wheel, the parameters of the inner structure in the first round of training are copied to the parameters of the updated outer structure. The data from different experience pool \mathbb{B}_x are sampled, which means that the environmental conditions have changed. The time complexity of the code is determined by m episodes, n MDs and t steps. Since the maximum step is 50, which means that all tasks can be trained in each episode, the time complexity of the algorithm is $O(mn)$. The space complexity of the code is mainly determined by the experience stored in the experience pool. The size of these experiences depends on the number of MDs, so the space complexity of the algorithm is $O(m)$. The pseudocode of Meta-MADDPG algorithm is presented as follows:

Algorithm 1 Meta-MADDPG method

Initialize actor network $\mu(s|\theta^\mu)$ and critic network $Q(s, a|\theta^Q)$ with weights θ^μ, θ^Q .
Initialize actor network $\mu_{\text{meta}}(s|\theta^{\mu_{\text{meta}}})$ and critic network $Q_{\text{meta}}(s, a|\theta^{Q_{\text{meta}}})$.
Initialize target network μ' and Q' with weights $\theta^{\mu'} \leftarrow \theta^\mu, \theta^{Q'} \leftarrow \theta^Q$
Initialize experience pool \mathbb{B}_x

for episode = 1, T **do**
 Reset environment
 if episode % $\psi = 0$ **do**
 $x \leftarrow x + 1$
 Set \mathbb{B}_x as experience pool
 Reset a different environment where some of the conditions have changed
 Initialize a random process \mathcal{N} for action exploration
 for $n = 1, N$ **do**
 Receive initial observation state s_n
 Select action $a_n = \mu(s_n|\theta^\mu) + \mathcal{N}_n$
 Execute action a_n and observe reward r_n , next state s_{n+1} and whether done d_n
 Store transition $(s_n, a_n, r_n, s_{n+1}, d_n)$ in \mathbb{B}_x
 Sample a random minibatch of S transition $(s^j, a^j, r^j, s'^j, d^j)$ from \mathbb{B}_x
 If episode % $\psi = 0$ **and** $n = 1$ **do**
 Update parameters $\theta^{Q_{\text{meta}}}$ by minimizing the loss L_{sum} average
 Update parameters $\theta^{\mu_{\text{meta}}}$ using the sampled gradient
 Copy meta parameters \rightarrow online network parameters:
 $\theta^Q \leftarrow \theta^{Q_{\text{meta}}}$
 $\theta^\mu \leftarrow \theta^{\mu_{\text{meta}}}$
 for $i = 1, M$ **do**
 Set $y_i = r_i^j + (1 - d_i^j)YQ'(s'^j, a_1^j, \dots, a_M^j)|_{a_k^j = \mu_k^j(s_k^j)}$
 Calculate loss $\mathcal{L}(\theta_i) = \frac{1}{S} \sum_i (y_i - Q_i^\mu(s^j, a_1^j, \dots, a_M^j))^2$
 Update critic online network parameters θ^Q by minimizing the loss
 Update the policy online network parameters θ^μ using the sampled gradient:

$$\nabla_{\theta^\mu} \mu|_{s_i} \approx \frac{1}{S} \sum_j \nabla_{\theta_i \mu_i}(s_i^j) \nabla_{a_i} Q_i^\mu(s^j, a_1^j, \dots, a_i, \dots, a_M^j)$$

 Soft update the networks:

$$\theta^{Q'} \leftarrow \lambda \theta^Q + (1 - \lambda) \theta^{Q'}$$

$$\theta^{\mu'} \leftarrow \lambda \theta^\mu + (1 - \lambda) \theta^{\mu'}$$

 Re-sample and calculate loss L_{meta} and cumulative
 end for
 end for
end for

5 Evaluation

In this section, we present simulation results to evaluate the performance of the algorithm presented in this paper. We set the base station in the center of the MEC system, and

the MEC server deployed on the base station with computing power $f^{ES} = 10$ GHz/s and bandwidth $B = 10$ MHz. Mobile devices move randomly in the MEC system. The computing power of each mobile device is $f^{MD} = 1$ GHz/s, the transmission power is $p^{MD} = 900$ mW, and the idle power is $p^{idle} = 100$ mW [15]. We set D_n^m (in kbits) as the random value between (400,500), Z_n^m (in Megacycles) as the random value between (800,900), and τ^m (in second) as the random value between (0.8,0.9). We set up five sets of environment configurations, as shown in Table 1.

Table 1. The experimental environment configurations

Configuration number	The experimental environment configurations		
	Number of MDs	BS computational capacity	Bandwidth
1	10	1×10^{10}	1×10^7
2	20	2×10^{10}	2×10^7
3	30	3×10^{10}	3×10^7
4	40	4×10^{10}	4×10^7
5	50	5×10^{10}	5×10^7

We compared the proposed algorithm with DQN, MADDPG, and Full-Local in various configurations. Full-Local indicates that all mobile devices execute their computing tasks locally. The ordinate cost represents the weighted sum of total energy consumption and total delay.

In Fig. 3, we run several algorithms under the five configurations shown in Table 1. As the numbers of MD, base station computing power and bandwidth increase in the same proportion, the average cost of Meta-MADDPG decreases naturally. Although the average cost of DQN and MADDPG decreases generally, the average cost of DQN and MADDPG increases when the number of MD is 20 because the MEC system resources in configuration 2 are too strained.

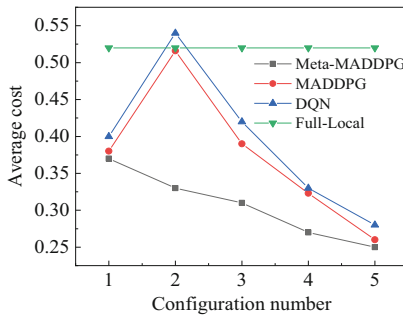


Fig. 3. Performance evaluation of average cost with the change of configuration.

In Fig. 4, we take the number of MD as a variable, and it can be seen that with the increase of MD, the average costs of all three algorithms, Meta-MADDPG, DQN and MADDPG, show upward trends. This indicates that the limited computing resources of MEC server are incapable of supporting the computation offloading of all MD. When the number of MD is 30, the average cost of DQN is equal to that of Full Local. The observation shows that DQN can achieve the minimum cost by maximizing local computing under the current environment configuration. This phenomenon shows the effect of base station computing power and bandwidth resources in the DQN model.

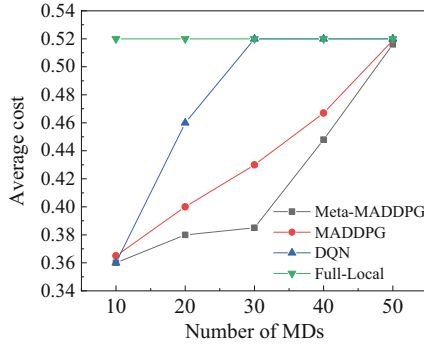


Fig. 4. Performance evaluation of average cost with the increase of number of MD.

Figure 5 shows the changing trend of average cost with the increase of bandwidth, and the three algorithms all decline naturally. This is because the increase in bandwidth cut down the time for MD offloading tasks to the MEC server, thus reducing the average cost of the MEC system. The Meta-MADDPG algorithm proposed in this paper gets the best results. When the bandwidth is increased to 5×10^7 , the performance of the three algorithms, except Full Local, is almost equal. The reason is that the bandwidth has been maximized in reducing the average cost and other factors are in play.

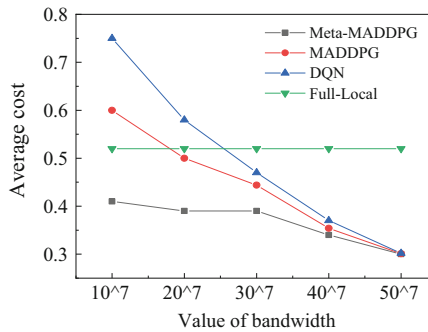


Fig. 5. Performance evaluation of average cost with the increase of bandwidth.

Figure 6 shows how Meta-MADDPG and MADDPG adapt to suddenly changing environments. The abscissa in Fig. 6 represents 1000 episodes in the MEC environment, that is, after the three elements of the task have been changed in a large span, while the ordinate represents the number of tasks completed in each episode. We consider the completion of 50 tasks as a strategy to find the optimal solution. It can be seen that the MEC system can complete all MD tasks within 50 time slot in Meta-MADDPG at 150 steps, and the scheduled tasks can be completed in most episodes after that. MADDPG was slow to complete a scheduled goal in the face of a suddenly changing environment while Meta-MADDPG showed better adaptability and faster convergence.

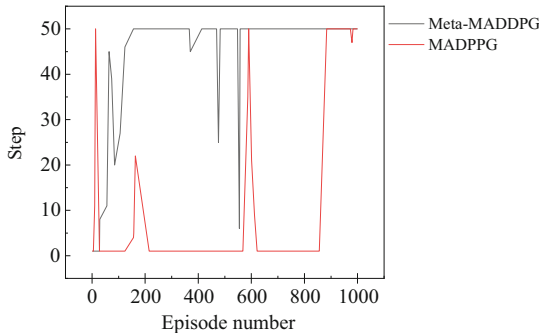


Fig. 6. Adapt to suddenly changing environments.

6 Conclusion

This paper proposes a MADDPG model of initialization parameters based on deep reinforcement learning. We conduct simulation experiments to compare this model with some other benchmark schemes. The MEC environment changes due to the large span change of task data size required computing power and delay sensitivity, which leads to the increased cost of scheduling decisions with weak fast adaptation ability during the convergence process, and even the tasks in the time slot will not be completed in the training process. Simulation results show that under different configuration parameters, the proposed scheme saves more energy and boasts more robust adaptability to new environments. Further work is underway to explore a more complex MEC system model that includes a three-tier network structure.

References

1. Sabella, D., Vaillant, A., Kuure, P., Rauschenbach, U., Giust, F.: Mobile edge computing architecture: the role of MEC in the internet of things. *IEEE Consum. Electr. Mag.* **5**(4), 84–91 (2016)
2. Mach, P., Becvar, Z.: Mobile edge computing: a survey on architecture and computation offloading. *IEEE Commun. Surv. Tutorials* **19**(3), 1628–1656 (2017)

3. Arulkumar, K., Deisenroth, M.P., Brundage, M., Bharath, A.A.: Deep reinforcement learning: a brief survey. *IEEE Signal Process. Mag.* **34**(6), 26–38 (2017)
4. Li, J., Gao, H., Lv, T., Lu, Y.: Deep reinforcement learning based computation offloading and resource allocation for MEC. In: *Proceedings of the 2018 IEEE Wireless Communications and Networking Conference* (2018)
5. Wang, Y., Ru, Z., Wang, K., Huang, P.: Joint deployment and task scheduling optimization for large-scale mobile users in multi-UAV-enabled mobile edge computing. *IEEE Trans. Cybern.* **50**(9), 3984–3997 (2020)
6. Peng, H., Shen, X.: Multi-agent reinforcement learning based resource management in MEC- and UAV-assisted vehicular networks. *IEEE J. Sel. Areas Commun.* **39**(1), 131–141 (2021)
7. Vilalta, R., Drissi, Y.: A perspective view and survey of meta-learning. *Artif. Intell. Rev.* **18**, 77–95 (2002)
8. R. Lowe, Y. Wu, A. Tamar, J Harb, OpenAI Pieter Abbeel, Igor Mordatch: Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. *Advances in Neural Information Processing Systems* 30 (NIPS 2017)
9. Duan, Y., Schulman, J., Chen, X., et al.: RL [^] 2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint [arXiv:1611.02779](https://arxiv.org/abs/1611.02779)* (2016)
10. Wang, C., Yu, F.R., Liang, C., Chen, Q., Tang, L.: Joint computation offloading and interference management in wireless cellular networks with mobile edge computing. *IEEE Trans. Veh. Technol.* **66**(8), 7432–7445 (2017)
11. Wen, Y., Zhang, W., Luo, H.: Energy-optimal mobile application execution: taming resource-poor mobile devices with cloud clones. In: *Proceedings of the 2012 IEEE INFOCOM* (2012)
12. Zhou, P., Finley, B., Li, X., Tarkoma, S., Kangasharju, J., Ammar, M., Hui, P.: 5G MEC computation handoff for mobile augmented reality. *Computer Science Networking and Internet Architecture* (2021)
13. Hochba, D.S.: Approximation algorithms for NP-hard problems. *ACM SIGACT News* **28**(2), 40–52 (1997)
14. Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement learning: a survey. *J. Artif. Intell. Res.* **4**, 237–285 (1996)
15. Cao, Y., Jiang, T., Wang, C.: Optimal radio resource allocation for mobile task offloading in cellular networks. *IEEE Network* **28**(5), 68–73 (2014)



An Evolutionary Game Based Computation Offloading for an UAV Network in MEC

Qi Gu and Bo Shen^(✉)

School of Computer Science, Northwestern Polytechnical University,
Xi'an 710129, China
shen@nwpu.edu.cn

Abstract. With the rapid development of information technology, low-cost unmanned aerial vehicles (UAVs) appear. With advanced sensing and actuating technologies, they are being increasingly applied to a variety of scenarios. However, considering their limited computing resource and restricted battery capability, the computation-intensive tasks or data-intensive tasks will face tough challenges. With the aid of Mobile Edge Computing (MEC), moving computation-intensive tasks from resource-constrained UAVs to edge cloud servers can significantly save energy and finally achieve impressive performance. This paper proposes an evolutionary game based algorithm to solve the computation offloading problem for UAVs. By replicator dynamics, UAVs select the suitable service provider to offload the computation tasks via achieving a tradeoff between time delay, energy consumption and monetary cost when network externality exists. Simulation results show that the proposed algorithm can rapidly converge to evolutionary equilibrium and achieve desirable performance.

Keywords: Evolutionary game · Replicator dynamics · Computation offloading · Unmanned aerial vehicles · Mobile edge computing

1 Introduction

With the research and development (R&D) of drone technology, the power of unmanned aerial vehicles (UAVs), also known as drones, has been enhanced dramatically. Meanwhile, their cost has been dropped. As a consequence, UAVs can not only serve as aerial working platforms but also play increasingly important roles in military, commercial and civilian areas such as law enforcement by the police, environmental monitoring, geological survey, prevention of forest fires, movie and television photography, surveillance and so on. In many scenarios, UAVs are brought to deal with the specific tasks, such as video pre-processing, pattern recognition and feature extraction. These kinds of tasks typically require executing complex algorithms or running big models, which can

This work is supported by the National Natural Science Foundation of China under Grant no. 61902295.

be computation-intensive and requires dedicated and powerful processors [1]. Although the technologies of drones have been improved a lot, the limited computing capabilities present a major challenge for real-time data processing and decision-making. Besides, the constrained energy supply needs to be applied to the take-off, flying, hovering, landing. Performing highly intensive computation tasks onboard deteriorate the battery lifetime, which may further affect mission success. In order to address these challenges, offloading computation-intensive tasks to remote servers is widely regarded as a promising solution.

Mobile Edge Computing (MEC) has been introduced to bring computing and storage resources in physical proximity of end devices [2]. MEC helps to provide low-latency services to end devices which require intensive computation or a large volume of data [3]. Works on computation offloading in MEC have gained many advances [5]. In the scenarios where multiple edge servers co-exist in the same area for UAVs to select, choosing the most suitable edge server is a challenging problem. It is hard to make decision in a distributed manner. Game theory is expected as an effective method to solve the problem of resource allocation and multiple access in the distributed manner. The work presented in [1] adopted a game theory model in the network with three possible strategies to minimize the cost function. In [4], dynamic evolutionary game-based network-selection algorithms in the heterogeneous wireless networks are presented. Game theoretic approach for the computation offloading decision-making problem among multiple mobile device users for mobile-edge cloud computing are proposed in [6]. In order to achieve an efficient computation offloading coordination among end users, a decentralized mechanism was designed using potential game in [7]. The work in [8] proposed computation offloading game strategies for UAVs. Aiming at maximizing servers' utility and minimizing the cost of users, a game theoretic approach is proposed for jointly allocate wireless and cloud resources in [9].

Cost is the key when decisions are making. The potential factors evolving in cost function include delay, energy consumption and monetary cost. In order to solve the problem and select the most suitable providers to minimize cost, the paper considers the network externality. The number of UAVs sharing the same server is included in monetary cost. When the number of sharing UAVs exceed a threshold, time delay, energy consumption and monetary cost are positive correlation with it. Furthermore, the numerical normalization operation is introduced to maintain roughly the same influence for all three factors. Then a computation offloading algorithm based on evolutionary game (EG) is proposed. Simulation results shows the proposed method can converge to evolutionary equilibrium (EE) rapidly and achieves desirable performance when multiple service providers co-exist in the same area.

2 System Model

2.1 System Description

In the scenario, the heterogeneous network includes one macro base station (MBS) and K small base stations (SBSs). The MBS is locating in the center

of the area. The SBSs are scattered in the coverage area of MBS, following a poisson point distribution. Each base station (BS) connects to a small data center which provides edge computing resources. In the following, the MBS/SBS and its connecting server are referred as macro/small service provider (MSP/SSP).

As shown in Fig. 1, \mathcal{I} UAVs are hovering in the area. K SSPs overlaid with one MSP providing communication and computation services to UAVs. $\mathcal{K} = \{0, 1, 2, \dots, K\}$ represents the set of MSP and SSPs, where 0 represent the MSP and $\{1, 2, \dots, K\}$ the SSPs. We denote the set of service areas by $\mathcal{J} = \{1, 2, \dots, J\}$ in which service area is covered by only a macro cell or a macro cell and several small cells. The number of UAVs in service area j ($j \in \{1, 2, \dots, J\}$) is denoted as N_j , $\sum_{j=0}^J N_j = \mathcal{I}$.

In this model, we have similar assumptions in [12] with some differences.

- An UAV with multiple radio transceivers is capable of connecting to different SPs, and we assumes that the UAV can select only one SP for wireless transmission, resource scheduling and task offloading at the same time.
- The service area of each SP is a circular region that the SP is in the center. The coverage vary from SP to SP in view of their communication and computation capabilities.
- There are overlapped zones between two or more service areas of SPs. UAVs which are hovering in the overlapped zones could select different SPs according to their choices. In Fig. 1, area 1 only has MSP coverage. UAVs in the area can only subscribe to the MSP. The area 2 is covered by MSP and SSP6. UAVs in this area can select MSP or SSP6. However, one SP cannot located in the coverage of another except a SSP within the range of MSP.

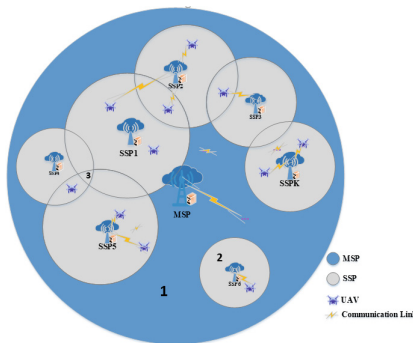


Fig. 1. Service areas considered in the mobile edge computing environment.

Each UAV has some computational tasks to be executed. Tasks are characterized by C_i , the number of instructions to be executed, and D_i , the data size of the tasks. The available spectrum bandwidth of SP k is W_k . It can process tasks from all UAVs connecting to it concurrently with the computing capacity

of F_k instructions per unit time. The number of UAVs connecting to SP k is noted as n_k . The bandwidth and computing capability of each SP are averagely allocated to all UAVs connecting to it which are W_k/n_k and F_k/n_k .

We consider a channel model similar to [10] and [13]. The orthogonal sub-channels are assigned to the UAVs. Therefore, there is no channel interference. The channel between each SP and its connected UAV is composed of a fixed distance path-loss, a slowly component modeled by lognormal shadowing and Rayleigh fast fading with unit average power. Each SP allocates rate adaptively depends on the received signal to noise ratio (SNR) per drone. As SSPs and MSP averagely allocate the bandwidth to the subscribers, all UAVs that subscribe to the same SP share the long-term expected throughput equally. η_k denotes the throughput of SP k in bit/s/Hz. Therefore, the expected throughput R_k for the UAV which subscribe to SP k can be described as $R_k = (W_k\eta_k)/n_k$.

2.2 Cost Design

Time Delay. According to previous works, the delay due to task offloading is composed of four parts: $T_i = \Delta_i^{ul} + \Delta_i^{dl} + \Delta_i^{bl} + \Delta_i^{exe}$, the uplink communication delay Δ_i^{ul} , the downlink delay Δ_i^{dl} , the backhaul link delay Δ_i^{bl} and the task processing delay Δ_i^{exe} . However, compared with the size of input data D_i , the size of output data is so small that could be neglected. Furthermore, the data rate of backhaul link between BSs and their corresponding servers is high. The distance between them is also small. Therefore, Δ_i^{bl} is too small to be considered. Hence, the delay is computed as: $T_i = \Delta_i^{ul} + \Delta_i^{exe}$. Δ_i^{ul} is given as $\Delta_i^{ul} = D_i/r_i$ where $r_i = R_k = (W_k\eta_k)/n_k = w_i\eta_k$. Δ_i^{exe} is defined as $\Delta_i^{exe} = C_i/f_i$ where $f_i = F_k/n_k$, if UAV i select BS k . Then the time delay can be also written as:

$$T_i = D_i/(w_i\eta_k) + C_i/f_i \quad (1)$$

Energy Consumption. The energy consumption of UAV i is $E_i = p_i \Delta_i^{ul} + p_i^r \Delta_i^{dl}$, which includes both uplink and downlink energy consumption, where p_i represents the transmission power of UAV i which is fixed. p_i^r denotes the received power of UAV i . As mentioned above, the energy consumption of data receiving stage is neglected. The energy consumption is given by:

$$E_i = p_i \Delta_i^{ul} = p_i D_i/(w_i\eta_k) \quad (2)$$

Monetary Cost. In this paper, the monetary cost of UAV i is composed of three parts: the communication cost, the computation cost and the basic cost which is related to the number of UAVs sharing same SP:

$$M_i = q_k w_i \eta_k + g_k f_i + lp_k n_k \quad (3)$$

where q_k and g_k mean the price of per unit transmission bandwidth and the price of per 100 unit computation resources of SP k , respectively. lp_k is the basic cost related to n_k . When the number of UAVs n_k is larger than a certain value,

the monetary cost M_i will increase as n_k goes up. In the other case, it shows M_i is inversely proportional to n_k . These kind of monetary cost lead UAVs to choose SPs rationally.

Additionally, we found that monetary cost M_i is much larger than time delay cost T_i and energy consumption cost E_i , if use simulation data to calculate with the above formulas. In order to set the three parts of cost to have the influence with similar scale, the numerical normalization is carried out: the monetary cost M_i 100 is set to one percent of the original value.

Cost Function. The cost function is defined as the weighted sum of delay, energy consumption and monetary cost:

$$Z_i = \alpha T_i + \beta E_i + \gamma M_i \quad (\alpha + \beta + \gamma = 1) \quad (4)$$

α , β and γ mean the weighted parameters of delay, energy and monetary cost, satisfying $\alpha + \beta + \gamma = 1$. Then the total cost of all UAVs could be expressed as:

$$\min \sum_j^J \sum_i^{N_j} Z_i \quad (5)$$

The objective of the system model is to optimize the function, i.e., to minimize the total cost.

3 Evolutionary Game Formulation

3.1 Game Formulation

The evolutionary game in the MEC-aided heterogeneous environment can be described as follows:

- Players: In each service area, each UAV that could select SPs is a player of the game.
- Population: The population in this evolutionary game refers to the set of UAVs in the same service area, expressed as $\mathcal{J} = \{1, 2, \dots, J\}$ and J is the number of populations. Besides, N_j is the number of UAVs in population j , satisfying $\sum_{j=1}^J N_j = \mathcal{I}$.
- Strategy: The strategy of each player refers to the selection of SPs, denoted as $\mathcal{K} = \{0, 1, 2, \dots, K\}$ where 0 represent the MSP and $\{1, 2, \dots, K\}$ is the SSPs.
- Population share: The amount of UAVs that select SP k in the population j is represented by n_k^j . Then the population share of the UAVs that select SP k in the population j is $x_k^j = n_k^j / N_j$, where $x_k^j \in (0, 1]$.
- Population state: The population shares of all SPs in the population j constitute the population state, denoted by the vector $x_j = [x_0^j, x_1^j, x_2^j, \dots, x_K^j]$, $\sum_{k=0}^K x_k^j = 1$. We use matrix \mathcal{X} to denote the population state space, which contains the states of all J populations.

- Cost function: As mentioned above, we use the cost function to quantify the consumption of an UAV when the task is offloaded. For a particular population j , the cost of UAV i selecting SP k can be written as: $Z_k^j(i) = \alpha T_i + \beta E_i + \gamma M_i$. In the case that all UAVs specify the same size of tasks and weights, $Z_k^j(i)$ can be noted as Z_k^j .

3.2 Replicator Dynamics

In evolutionary game, replicator dynamics provides an approach to converge to the equilibrium selection. The replicator dynamics equation is defined as follows:

$$\dot{x}_k^j(t) = \sigma x_k^j(t)[Z_k^j(t) - \bar{Z}_j(t)] \tag{6}$$

where σ is the gain parameter, satisfying $\sigma > 0$. It is the speed at which the player adjusts its strategy. $Z_k^j(t)$ is the current cost of the players choosing strategy k in population j . $\bar{Z}_j(t)$ is the average cost of the population j , which can be derived as:

$$\bar{Z}_j(t) = \sum_{k=0}^K (x_k^j(t) Z_k^j(t)) \tag{7}$$

Via the replicator dynamics, the amount of UAVs selecting SP k will increase if the cost of selecting SP k is below the average cost. Otherwise, UAV will not select SP k . This replicator dynamics satisfies the condition of $\sum_{k=0}^K \dot{x}_k^j(t) = 0$.

3.3 Evolutionary Equilibrium

The evolutionary stable strategy (ESS) is the basic equilibrium in evolutionary game theory. If ESS is reached, it is impossible for a small mutant population to invade the population. The population share x will not change unless there is a strong external impact [10].

Supposing that there is a small portion of players switching from the equilibrium strategy to a different strategy y . This part of players is viewed as mutants of the population. Its size can be expressed as a normalized value $\epsilon \in (0, 1)$. Then the population state after mutation is $(1 - \epsilon)x + \epsilon y$.

With the above notations, ESS is described as follows:

If a strategy \mathcal{X}^* is an ESS in population j , for any $y_j \neq x_j$, there is some $\epsilon_y \in (0, 1)$ such that for all $\epsilon \in \epsilon_y$, the following inequality holds:

$$Z(x_j, (1 - \epsilon)x_j + \epsilon y_j) < Z(y_j, (1 - \epsilon)x_j + \epsilon y_j) \quad \forall y_j \in \mathcal{J} \tag{8}$$

where $Z(x_j, (1 - \epsilon)x_j + \epsilon y_j)$ and $Z(y_j, (1 - \epsilon)x_j + \epsilon y_j)$ are the expected costs of non-mutants and mutants of population j , respectively. The $\epsilon_y \in (0, 1)$ can be regarded as an invasion barrier which represents the maximum proportion of players using mutant strategies. ϵ_y is resisted by the ESS, satisfying the larger ϵ_y is, the more robust ESS is [13].

3.4 Offloading Algorithm Description

The objective is to minimize the total cost. According evolutionary game, reaching the EE leads to the optimization goal. In order to reach the EE, the player who is choosing the SP with higher cost will change to select a lower SP repeatedly. The SPs will allocate their communication and computation resources to UAVs averagely. The proposed algorithm based on replicator dynamics can be described as Algorithm 1.

Algorithm 1. Evolutionary game algorithm

Require: \mathcal{J} : the set of populations; J : the amount of populations; N_j : the amount of UAVs of population j ; \mathcal{K} : the set of SPs; K : the amount of SPs; C : computing requirement; D : input date size; W_k : the bandwidth of SP k ; F_k : the computing capacity of SP k ; η_k : the throughput of SP k ;

- 1: Initialize: $\mathcal{J}, J, \mathcal{K}, K, N_j, C, D, W_k, F_k, \eta_k, \alpha, \beta, \gamma, p, q, g, lp$.
- 2: **for** $j = 1 : J$ **do**
- 3: **for** $i = 1 : N_j$ **do**
- 4: UAV i in population j randomly choose accessible SP k
- 5: **end for**
- 6: **end for**
- 7: **loop**
- 8: SP k acquires n_k and allocates $W_k F_k$ averagely
- 9: Computes Z_k^j and \bar{Z}_j
- 10: **if** $N_j \neq 0$ **then**
- 11: **for** $i = 1 : N_j$ **do**
- 12: **if** $Z_k^j(i) > \bar{Z}_j$ **then**
- 13: **if** $(Z_k^j(i) - \bar{Z}_j) / \bar{Z}_j > rand()$ **then**
- 14: choose strategy t , where $t \neq i$ and $Z_k^j(t) < Z_k^j(i)$
- 15: **end if**
- 16: **end if**
- 17: **end for**
- 18: **end if**
- 19: **end loop** for all players in all populations.

4 Simulation Results

4.1 Parameter Settings

The simulation scenario is shown in Fig. 2. We consider a heterogeneous network with 1 MSP and 3 SSPs. Just like the assumptions and requirements of Sub-sect. 2.1 System Model, the MSP lies in the center of the area, SSPs are scatterd following poisson point distribution, UAVs are randomly created within the coverage of SPs.

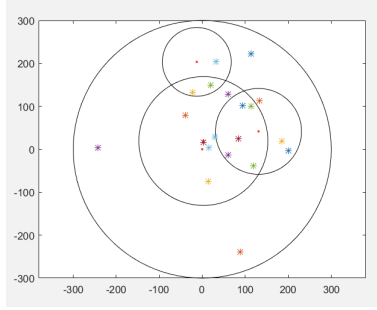


Fig. 2. Simulation result of service area.

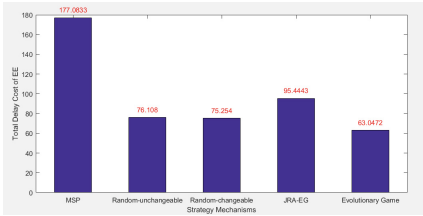


Fig. 3. Total delay of different strategies.

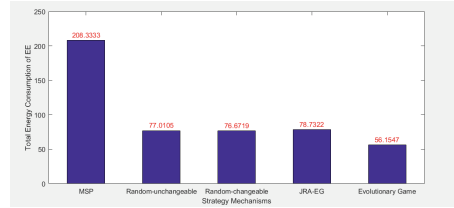


Fig. 4. Total energy consumption of different strategies.

The radius of the service areas of SPs is set to $sp_r = [300,150,100,80]$ m. As shown in Fig. 2, in the service scenario up to 8 populations could be formed, i.e., $J = 8$. The total amount of UAVs are 25. The default cost function weight is set to $\alpha = 0.4$, $\beta = 0.3$ and $\gamma = 0.3$. The bandwidth and computation capacity of SPs are set to $W = [200,150,100,80]$ MHz, $F = [80000,60000,50000,40000]$ Mega/s, respectively. The throughput of the SPs is set to $\eta = [1.5,1.7,1.8,2]$ bit/s/Hz, and the transmit power of the UAVs is $p = [10,8,6,5]$ mW. The price for communication resources is $q = [0.05,0.035,0.02,0.01]$ \$/Mbit. The charge for computation resource is $g = 0.1$ \$/100 Mega, and the basic cost $lp = 2$ \$ per UAV. For the computing task, we use the face recognition program, which the data size for the computation offloading $D = 10$ M and the total number of instructions $C = 20000$ Megacycles.

4.2 Performance Evaluation

Overhead of Different Strategy Mechanisms. Figure 3 and Fig. 4 respectively reveals the total delay and total energy consumption of different strategies of SP selecting. The MSP strategy is only choosing MSP. Random-unchangeable is the strategy that players select the SPs randomly at the beginning and will not change during the system-running. In Random-changeable, players can select their SPs randomly at every slot. The JRA-EG is the proposed strategy in [10].

The evolutionary game strategy is the one proposed by the paper. The results are averaged by 10^5 times simulations. By comparing the MSP with the other four, we can see that pure selection of MSP leads to the highest cost: the more players subscribe to the same SP, the higher the cost of the SP. Furthermore, the Random-unchangeable and the Random-changeable have similar cost, as both them are randomly selecting strategies. In the simulation, EG could guarantee the lowest total cost. This is due to the fact that players always choose the cheaper strategy to decrease the cost in every iteration until reaching EE, at which point players could not lower cost through changing their strategies. And the proposed monetary cost function and the numerical normalization operation improve the superiority of the algorithm.

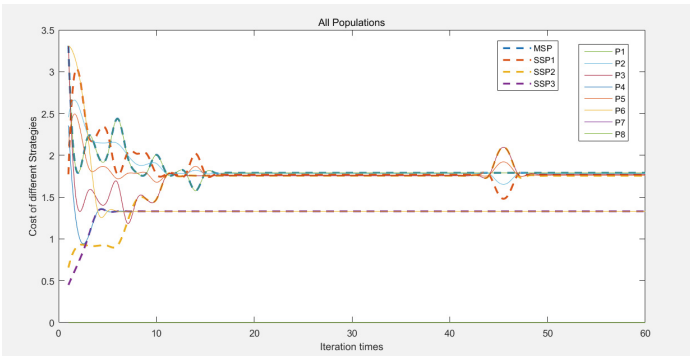


Fig. 5. Cost of different SPs and population.

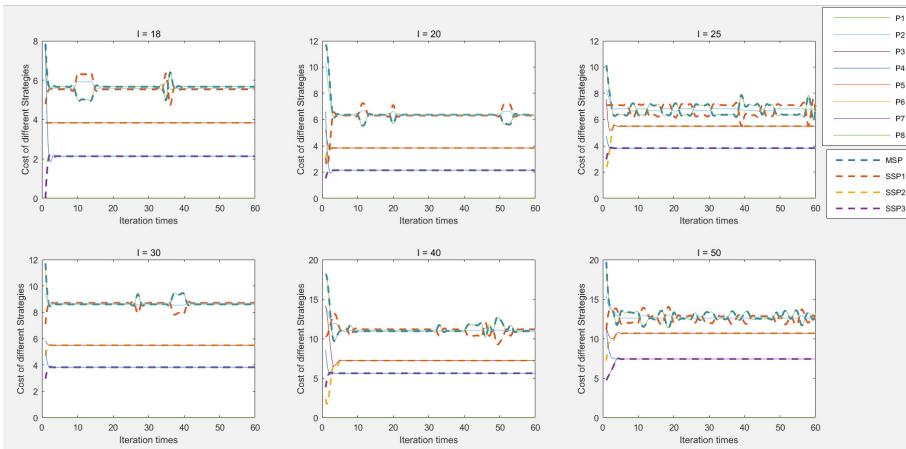


Fig. 6. Convergence with different amount of UAVs.

Evolutionary Equilibrium. According to the replicator dynamics, the amount of the subscribers of the SPs which have lower cost than average will increase. In the cost design, the cost will increase subsequently. Correspondingly, the amount of UAVs choosing SPs decrease if the cost is higher than the average. Then the cost will decrease. As shown in Fig. 5, the dotted lines represent the cost trends of 4 SPs, while the solid lines represent the average cost of 8 populations. As the iterations evolve, the overhead of selecting different SPs gradually converge to average cost. It is the equilibrium cost of all UAVs, at which UAVs have no motivation to change its strategy.

Convergence with Different Number of UAVs. Figure 6 shows the convergence of the evolutionary strategy when the amount of UAVs vary. We can see that the cost grows up as the amount of players increases. The results also show there are more fluctuations in the convergence process.

We analyze there are two main reasons that cause the fluctuations. Firstly, the cost of SPs are influenced by multiple populations. The amount of players in each population and the amount of populations are random distributed. The process of convergence is instable relatively. Secondly, the specific value of each parameter influence the process, too. However, from Fig. 7, we can conclude that the normal fluctuations during the iterative convergence process would not impede the convergence of the proposed strategy.

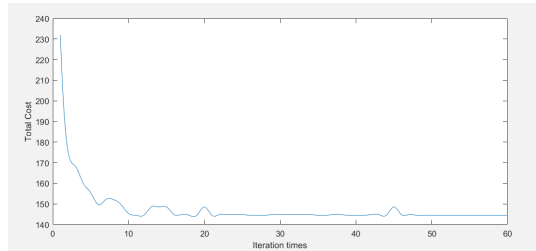


Fig. 7. Convergence of total cost.

Average Cost. The paper also analyzes the change of different input data and computing requirement to evaluate their impact on the average cost. Figure 8 and Fig. 9 shows that the average cost increases as the input data size or computing requirement increases. However, the influence of computing requirement is smaller compare with the effect of input data size. Part reason of the phenomenon is that input data size is related to both delay and energy consumption, which computing requirement only affects delay.

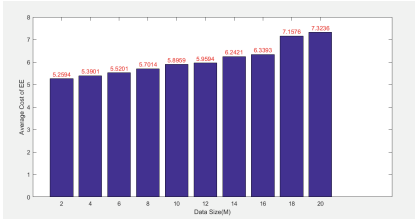


Fig. 8. Impact of different data size on the average cost.

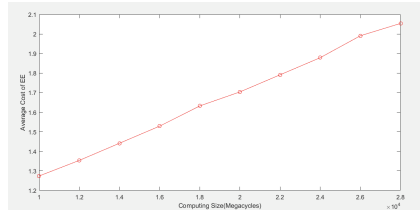


Fig. 9. Impact of different computing size on the average cost.

5 Conclusion

At this stage, we do not consider the strategy selection of UAVs when they are moving. In future, we will take into account the movement of UAVs. The strategy space of UAVs will dynamical change. Furthermore, we will study how to allocate the resources of SPs in the case that UAVs have different QoS requirements.

References

1. Messous, M.-A., Sedjelmaci, H., Houari, N., Senouci, S.-M.: Computation offloading game for an UAV network in mobile edge computing. In: IEEE International Conference on Communications (ICC), pp. 1–6 (2017)
2. Mao, Y., You, C., Zhang, J., Huang, K., Letaief, K.B.: A survey on mobile edge computing: the communication perspective. *IEEE Commun. Surv. Tutor.* **19**(4), 2322–2358 (2017)
3. Mehrabi, M., You, D., Latzko, V., Salah, H., Reisslein, M., Fitzek, F.H.P.: Device-enhanced MEC: multi-access edge computing (MEC) aided by end device computation and caching: a survey. *IEEE Access* **7**, 166079–166108 (2019)
4. Niyato, D., Hossain, E.: Dynamics of network selection in heterogeneous wireless networks: an evolutionary game approach. *IEEE Trans. Veh. Technol.* **58**(4), 2008–2017 (2009)
5. Mach, P., Becvar, Z.: Mobile edge computing: a survey on architecture and computation offloading. *IEEE Commun. Surv. Tutor.* **19**(3), 1628–1656 (2017)
6. Chen, X., Jiao, L., Li, W., Fu, X.: Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM Trans. Netw.* **24**(5), 2795–2808 (2016)
7. Chen, X.: Decentralized computation offloading game for mobile cloud computing. *IEEE Trans. Parallel Distrib. Syst.* **26**(4), 974–983 (2015)
8. Messous, M., Senouci, S., Sedjelmaci, H., Cherkaoui, S.: A game theory based efficient computation offloading in an UAV network. *IEEE Trans. Veh. Technol.* **68**(5), 4964–4974 (2019)
9. Lan, Z., et al.: A hierarchical game for joint wireless and cloud resource allocation in mobile edge computing system. In: 2018 10th International Conference on Wireless Communications and Signal Processing (WCSP), pp. 1–7 (2018)
10. Zhang, J., et al.: An evolutionary game for joint wireless and cloud resource allocation in mobile edge computing. In: 2017 9th International Conference on Wireless Communications and Signal Processing (WCSP), pp. 1–6 (2017)

11. Dong, Y., Peng, Y., Guo, X., Chu, F., Zhang, L.: Offloading decision algorithm using evolutionary game for mobile edge computing. In: 2019 IEEE 2nd International Conference on Information Communication and Signal Processing (ICICSP), pp. 210–214 (2019)
12. Zhou, L., et al.: A dynamic graph-based scheduling and interference coordination approach in heterogeneous cellular networks. *IEEE Trans. Veh. Technol.* **65**(5), 3735–3748 (2016)
13. Zhu, K., Hossain, E., Niyato, D.: Pricing, spectrum sharing, and service selection in two-tier small cell networks: a hierarchical dynamic game approach. *IEEE Trans. Mob. Comput.* **13**(8), 1843–1856 (2014)



Edge Collaborative Task Scheduling and Resource Allocation Based on Deep Reinforcement Learning

Tianjian Chen¹, Zengwei Lyu^{1,3(✉)}, Xiaohui Yuan², Zhenchun Wei^{1,3(✉)},
Lei Shi^{1,3}, and Yuqi Fan^{1,3}

¹ School of Computer Science and Information Engineering,
Hefei University of Technology, Hefei 230009, China

² Department of Computer Science and Engineering University of North Texas
Denton, Denton, TX 76203, USA

³ Engineering Research Center of Safety Critical Industrial Measurement
and Control Technology, Ministry of Education, Hefei 230009, China
986397142@qq.com

Abstract. With the development of the sixth generation mobile network (6G), the arrival of the Internet of Everything (IoE) is accelerating. An edge computing network is an important network architecture to realize the IoE. Yet, allocating limited computing resources on the edge nodes is a significant challenge. This paper proposes a collaborative task scheduling framework for the computational resource allocation and task scheduling problems in edge computing. The framework focuses on bandwidth allocation to tasks and the designation of target servers. The problem is described as a Markov decision process (MDP). To minimize the task execution delay and user cost and improve the task success rate, we propose a Deep Reinforcement Learning (DRL) based method. In addition, we explore the problem of the hierarchical hash rate of servers in the network. The simulation results show that our proposed DRL-based task scheduling algorithm outperforms the baseline algorithms in terms of task success rate and system energy consumption. The hierarchical settings of the server's hash rate also show significant benefits in terms of improved task success rate and energy savings.

Keywords: Edge collaborative · Task scheduling · Deep reinforcement learning · Hierarchical server

1 Introduction

Field of metaverse and autonomous driving, massive amounts of data place incredibly high demands on hash rate, and the construction of a mobile edge computing (MEC) network platform is expected to cope with this challenge

Supported by the Natural Science Foundation of Anhui Province (2108085MF202).

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022

L. Wang et al. (Eds.): WASA 2022, LNCS 13473, pp. 598–606, 2022.

https://doi.org/10.1007/978-3-031-19211-1_49

[1, 2]. The network of edge nodes allocates computing resources upon requests of offloading a task to the network. Scheduling methods have been developed, e.g., workflow-based dynamic scheduling algorithms [3, 4], self-adaptive learning particle swarm optimization (SLPSO) [5, 6]. Methods based on Deep Reinforcement Learning are developed for the task scheduling problem [7] in the MEC system.

In addition, setting optimization goals is essential in algorithm evaluation. Many scholars use task delay [8] and energy consumption [9, 10] as evaluation goals [11–14].

This paper proposes an optimal scheduling strategy suitable for delay constraints and user rent constraints to address this challenge. We propose an improved DRL-based task scheduling algorithm to solve multiple tasks' edge cooperative scheduling problem. Our main contributions are as follows:

- An edge collaborative task scheduling framework is proposed for the computational resource allocation and task scheduling problems in edge scenarios. The problem considers the bandwidth resources of the server, task characteristics, task queue state of the target server, and arithmetic power level.
- We propose a DRL-based algorithm to solve this scheduling problem, improving algorithm convergence and minimizing the weighted sum of task latency and cost by allocating appropriate bandwidth resources and target servers.
- Simulation results demonstrate that the proposed method outperforms the baseline algorithms. In addition, the hierarchical settings of arithmetic power for servers in the network effectively reduces the system energy consumption.

The rest of this article is organized as follows. Section 2 introduces the system model and describes the problem. Section 3 presents our proposed algorithm including algorithm design and process description. Section 4 presents our experimental results and discussion. Section 5 concludes this paper with a summary.

2 System Model and Problem Description

2.1 Model Overview

As shown in Fig. 1, our model consists of servers with hierarchical hash rate and multiple end-users. There are two waiting queues on each server, the waiting scheduling queue, and the waiting execution queue. The edge servers can be represented as $\mathcal{M} = \{1, 2, \dots, M\}$. $\mathcal{N} = \{1, 2, \dots, N\}$ represents a task generated by the end-user.

Users send task requests to the closest server m' first, which we define as the local server. The local server receives the task $A_{m'n} = \{C_{m'n}, L_{m'n}, Cost_{m'n}\}$, where $C_{m'n}$ denotes the task size and $L_{m'n}, Cost_{m'n}$ denotes the task's latest response time and the maximum acceptable overhead for that task, respectively.

2.2 Task Scheduling Model

After the user sends a task request to the local server m' , the scheduler in the local server m' allocates transmission bandwidth and gives it to the specified target server m for execution. The entire scheduling process can be divided into the following four stages.

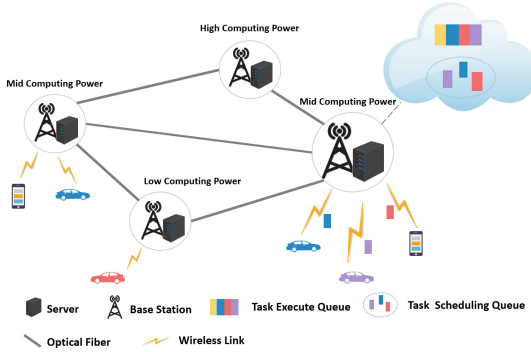


Fig. 1. The structure of our system model.

Task Transmission: After the local server m' receives the task request, it allocates appropriate bandwidth resources to it, and the end-user transmits the task to the local server m' through the wireless channel, we ignore the time delay, the uplink transmission rate can be defined as:

$$o_{m'n} = Q \log \left(1 + \frac{p_{m'n} |h_{m'n}|^2}{\sigma^2} \right) \tag{1}$$

where $p_{m'n}$ is transmission channel bandwidth between the task n and the local server m' , $h_{m'n}$ represents the channel gain, which is time-varying, and σ^2 is the noise. The transfer time of the task is:

$$ST_{m'n} = \frac{C_n}{o_{m'n}} \tag{2}$$

Task Scheduling: The scheduler of the local server gives the optimal task scheduling policy according to the currently observed network status, such as the bandwidth of the local server and the available computing resources of each server in the network.

Waiting for Execution: We use Que_m to indicate the number of tasks in the waiting execution queue of the target server m . Therefore, the waiting execution time of the task is:

$$WT_{mn} = \begin{cases} 0, & Que_m = 0 \\ freeT_{mn} - AT_{mn}, & else \end{cases} \tag{3}$$

where $freeT_{mn}$ represents the idle time of the server m after task n arrives. AT_{mn} is the task n arrival time to the target server. If it is in working condition, the task needs to wait. At this time, the idle time of the target server can be expressed as:

$$freeT_{mn} = \begin{cases} freeT_{mn'} + ET_{mn'}, & if\ freeT_{mn'} > AT_{n'} \\ AT_{n'} + ET_{mn'}, & else \end{cases} \tag{4}$$

where n' represents the previous task of the task n , $freeT_{mn'}$, $ET_{mn'}$, $AT_{n'}$ denote respectively the idle time of the target server after task n' arrive, the execution time of the task n' , and the arrival time after the task n' arrives at the target server.

Task Execution: The task size C_n and server compute speed f_m are known. So the estimated computing time of the task n on the server m can be expressed as

$$ET_{mn} = \frac{C_n}{f_m}. \quad (5)$$

In summary, after the completion of task n , the estimated completion time is the sum of the task n 's transmission time, waiting time for execution, and task execution time, which is expressed as follows

$$T_n = ST_{m'n} + WT_{mn} + ET_{mn}. \quad (6)$$

2.3 User Cost Model

The unit cycle price is u_m , and the higher the hash rate, the greater the u_m . Therefore, the total cost to be paid to the service after the execution of task n is

$$U_{mn} = \frac{C_n}{f_m} * u_m. \quad (7)$$

2.4 Problem Description

Our optimization goal is to minimize task execution delay and user cost overhead. The optimization problem can be described as follows

$$\begin{aligned} P1 : \min_p & \left\{ \alpha * \sum_{n=1}^N T_n + (1 - \alpha) * \sum_{n=1}^N U_{mn} \right\} \\ s.t. \quad C1 : & \alpha \in (0, 1) \\ C2 : & 0 < f_m \leq f_m^{max}, & \forall m \in \mathcal{M} \\ C3 : & 0 < p_{m'n} \leq p_{m'n}^{max}, & \forall m' \in \mathcal{M}, \forall n \in \mathcal{N} \\ C4 : & T_n \leq L_n, & \forall n \in \mathcal{N} \\ C5 : & U_{mn} \leq Cost_n, & \forall m \in \mathcal{M}, \forall n \in \mathcal{N} \end{aligned} \quad (8)$$

where α represents the weight factor, which the scheduler can define according to different task requirements [15]. $C2$ indicates that the server's hash rate is limited, and $C3$ indicates that the transmission power between each server and the task does not exceed the maximum value. Conditions $C4$ and $C5$ represent the task's constraints on time and cost.

3 Proposed Method

We transform Problem P1 into an MDP problem and design a Deep Deterministic Policy Gradient-based online scheduling and allocation (SA-DDPG) algorithm to solve it. The structure of our proposed network is shown in Fig. 2.

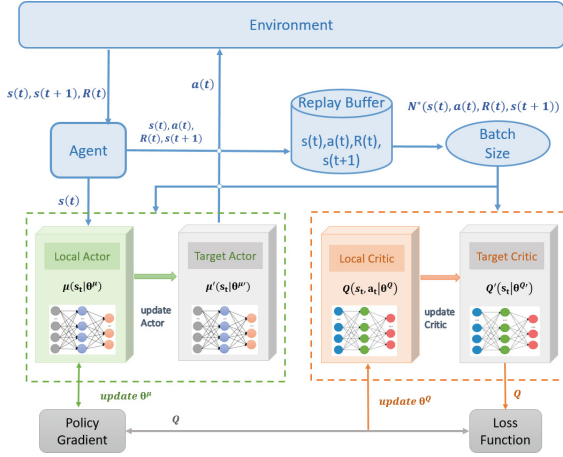


Fig. 2. SA-DDPG algorithm structure.

3.1 MDP-Based Task Scheduling Problems

State Space: The system state $s(t)$ consists of two components: the channel gain $h_{m'n}(t)$ between the user and the local server and the server state information $s_m(t)$ in the network. Thus, the state space at moment t can be expressed as :

$$s(t) = \{h_{m'n}(t), s_1(t), s_2(t), \dots, s_m(t)\} \tag{9}$$

Action Space: We define the action space as:

$$a(t) = \{p_{m'n}(t), a_{1n}(t), a_{2n}(t), \dots, a_{mn}(t)\} \tag{10}$$

Here $a_{mn} \in \{0, 1\}$, and $a_{mn} = 1$ represents offloading task n to server m , $a_{1n} + a_{2n} + \dots + a_{mn} = 1$.

Reward Function: Our optimization problem $P1$ is to minimize the delay and cost of task completion, so we set the reward to:

$$R = \frac{L_{m'n} - T_n}{L_{m'n}} + \frac{Cost_{m'n} - U_{mn}}{Cost_{m'n}} \tag{11}$$

3.2 SA-DDPG Algorithm Framework

We describe the network structure of the algorithm in Fig. 2. The weight parameters θ^μ and θ^Q of the Actor network and the Critic Network are randomly initialized at the beginning of the algorithm. We use Q' and μ' to improve learning stability in the target network.

Algorithm 1 Deep Deterministic Policy Gradient-based Online Scheduling and Allocation algorithm

Input: Task $A_{m'n} = \{C_{m'n}, L_{m'n}, Cost_{m'n}\}$, channel gain $h_{m'n}$
Output: Optimal scheduling policy $a^*(t)$
Initialize critic network $Q(s, a|\theta^Q)$ and actor network $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ ;
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$ and $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize the empty replay buffer \mathcal{B} ;
for $t = 1, 2, \dots, T$ **do**
Takes system state $s(t)$ as an input to the actor network and obtains action $a(t) = \mu(s(t)|\theta^\mu) + \delta_t$
Execute action $a(t)$, obtain the reward $R(t)$ and next state $s(t+1)$
Store transition tuple $s(t), a(t), R(t), s(t+1)$ into \mathcal{B}
if learning time reaches **then**
Agent collects K samples from \mathcal{B}
Update critic by minimizing the loss function in Equation (13)
Update actor policy by the deterministic policy gradient in Equation (14)
end if
regularly update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1 - \tau)\theta^{Q'}$$

$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1 - \tau)\theta^{\mu'}$$

end for

Algorithm Training: The Agent obtains states $s(t)$ from the environment and selects the current best action $a^*(t)$. After taking action $a^*(t)$, the Agent receives a reward R_t and subsequently observes the next state $s(t+1)$. The transition buffer is $\{s(t), a(t), R(t), s(t+1)\}$, which can be stored into the experience replay memory \mathcal{B} . In addition, the loss function in Fig. 2 is:

$$L = E \left[\left(R(t) + \gamma Q' \left(s(t+1), a(t+1) | \theta^{Q'} \right) - Q \left(s(t), a(t) | \theta^Q \right) \right)^2 \right] \quad (12)$$

where γ is the attenuation coefficient. And policy gradient can be expressed as:

$$\nabla_{\theta} \mathcal{T} = E \left[\nabla_a Q \left(s, a | \theta^Q \right) \Big|_{s=s(t), a=\mu(s(t))} \nabla_{\theta^\mu} \mu \left(s | \theta^\mu \right) \Big|_{s(t)} \right] \quad (13)$$

We formalize the SA-DDPG algorithm process in Algorithm 1.

4 Experimental Analysis

4.1 Settings

We compare the task success rate and energy consumption with the random scheduling algorithm, the round-robin [16] scheduling algorithm, the earliest scheduling algorithm, and the DRL algorithms of DQN [17] and DDQN. The detailed simulation parameters are shown in Table 1.

Table 1. Simulation Parameters

Parameters	Description	Value
f_m	Server hash rate	[12.5 MIPS/s,15 MIPS/s,17.5 MIPS/s]
C_n	Size of task n	[500 KB, 1200 KB]
M	Number of servers	15
$Mcapacity$	Computing capacity of the server	1000 MIPS
p_m^{max}	Maximum transmission power	0.2 M
ω	channel bandwidth	0.5 MHz
α_{actor}	Actor network learning rate	0.01
α_{critic}	Critic network learning rate	0.02
γ	The discount factor	0.9

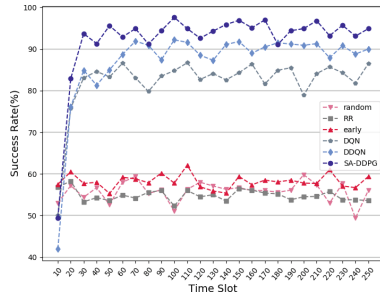


Fig. 3. The task success rate.

As shown in Fig. 3, the task success rate of the DRL-based family of algorithms is 30%-40% higher than that of the conventional algorithms because the DRL-based algorithms can make learning based on historical experience and continuously train to optimize the decisions.

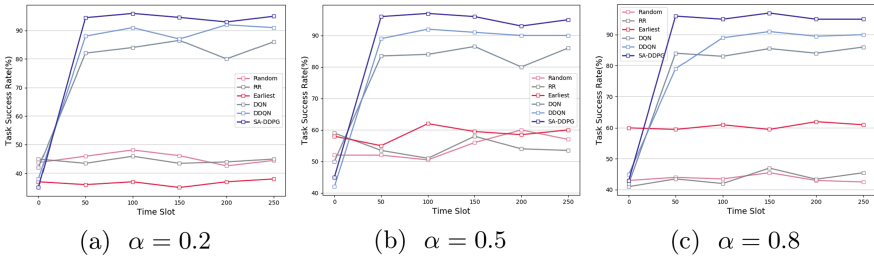


Fig. 4. Impact of different weighting factors α on task success rate.

We set α to 0.2, 0.5, and 0.8 for cost-sensitive tasks, balanced tasks, and delay-sensitive tasks, respectively. As shown in Fig. 4, the task success rate of our proposed algorithm consistently outperforms the other baseline algorithms in the three task type arrival scenarios, converging to about 93%.

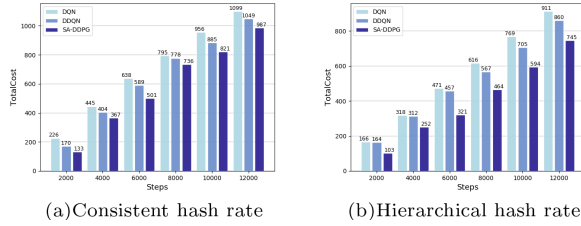


Fig. 5. Total energy consumption in different hash rate environments.

Furthermore, we explore the impact of the hierarchical hash rate of the servers on system energy consumption in Fig. 5. And after the server hash rate is hierarchical, the total energy consumption of all algorithms is reduced by about 25% compared with the ungraded case.

5 Conclusion

In this paper, we study the task scheduling problem in edge scenarios. To solve the problem of allocating bandwidth resources to servers and scheduling tasks among servers, we propose a DRL-based algorithm to reduce the total task latency and user overhead to maximize the success rate of tasks. In addition, we verified that our algorithm is highly adaptable and capable of handling any type of task. In future work, we can further explore mobility under multi-edge networks.

References

1. Kye, B., Han, N., Kim, E., Park, Y., Jo, S.: Educational applications of metaverse: possibilities and limitations. *J. Educ. Eval. Health Prof.* **18**, 32 (2021)
2. Abbas, M., Siddiqi, M.H., Khan, K., Zahra, K., Naqvi, A.U.: Haematological evaluation of sodium fluoride toxicity in oryctolagus cuniculus. *Toxicol. Rep.* **4**, 450–454 (2017)
3. Cai, Z., Li, X., Ruiz, R., Li, Q.: A delay-based dynamic scheduling algorithm for bag-of-task workflows with stochastic task execution times in clouds. *Futur. Gener. Comput. Syst.* **71**, 57–72 (2017)
4. Jiang, H., E, H., Song, M.: Dynamic scheduling of workflow for makespan and robustness improvement in the iaas cloud. *IEICE Trans. Inf. Syst.* **E100.D**(4), 813–821 (2017)
5. Zuo, X., Zhang, G., Tan, W.: Self-adaptive learning pso-based deadline constrained task scheduling for hybrid iaas cloud. *IEEE Trans. Autom. Sci. Eng.* **11**(2), 564–573 (2014)
6. Fu, Z., Tang, Z., Yang, L., Liu, C.: An optimal locality-aware task scheduling algorithm based on bipartite graph modelling for spark applications. *IEEE Trans. Parallel Distrib. Syst.* **31**(10), 2406–2420 (2020)

7. Yan, J., Bi, S., Zhang, Y.J.A.: Offloading and resource allocation with general task graph in mobile edge computing: a deep reinforcement learning approach. *IEEE Trans. Wireless Commun.* **19**(8), 5404–5419 (2020)
8. Du, J., Yu, F.R., Chu, X., Feng, J., Lu, G.: Computation offloading and resource allocation in vehicular networks based on dual-side cost minimization. *IEEE Trans. Veh. Technol.* **68**(2), 1079–1092 (2019)
9. Zhang, J., Hu, X., Ning, Z., Ngai, E.C.H., Zhou, L., Wei, J., Cheng, J., Hu, B.: Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks. *IEEE Internet Things J.* **5**(4), 2633–2645 (2018)
10. Hong, Z., Huang, H., Guo, S., Chen, W., Zheng, Z.: Qos-aware cooperative computation offloading for robot swarms in cloud robotics. *IEEE Trans. Veh. Technol.* **68**(4), 4027–4041 (2019)
11. Wang, Y., Sheng, M., Wang, X., Wang, L., Li, J.: Mobile-edge computing: partial computation offloading using dynamic voltage scaling. *IEEE Trans. Commun.*, 1 (2016)
12. Shah-Mansouri, H., Wong, V.W.S., Schober, R.: Joint optimal pricing and task scheduling in mobile cloud computing systems. *IEEE Trans. Wireless Commun.* **16**(8), 5218–5232 (2017)
13. Li, J., Qiu, M., Ming, Z., Quan, G., Qin, X., Gu, Z.: Online optimization for scheduling preemptable tasks on iaas cloud systems. *J. Parallel Distributed Comput.* **72**(5), 666–677 (2012)
14. Lu, H., He, X., Du, M., Ruan, X., Sun, Y., Wang, K.: Edge qoe: Computation offloading with deep reinforcement learning for internet of things. *IEEE Internet Things J.* **7**(10), 9255–9265 (2020)
15. Chun, B.G., Maniatis, P.: Augmented smartphone applications through clone cloud execution. In: *Proceedings of the 12th Conference on Hot Topics in Operating Systems, HotOS 2009*, p. 8. USENIX Association, USA (2009)
16. Devi, K., Paulraj, D., and B.M.: Deep learning based security model for cloud based task scheduling. *KSII Trans. Internet Inf. Syst.* **14**(9), 3663–3679 (2020)
17. Van Le, D., Tham, C.K.: A deep reinforcement learning based offload scheme in ad-hoc mobile clouds. In: *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 760–765 (2018)



Improving Gaming Experience with Dynamic Service Placement in Mobile Edge Computing

Yongqiang Gao^(✉) and Zheng Xu

Inner Mongolia University, Hohhot, China
gaoyongqiang@imu.edu.cn

Abstract. Mobile cloud gaming (MCG) can provide users with high-quality gaming services anytime, anywhere, but suffers from long network latency and huge wide-area traffic. In order to solve these problems, mobile edge computing (MEC) is envisioned as a promising approach to enable relevant computing at the edge. Since the quality of experience (QoE) of the game requires high frame rates and low network latency, the placement of service entities can affect the performance of MEC-enabled MCG. In addition, users have a high degree of mobility while enjoying MCG, so service migration is proposed to reduce QoE impairment, and service migration means an increase in system cost. To address these challenges, we investigate the service placement of MEC-enabled MCG. Considering the dynamics of the system, we propose to minimize the QoE impairment according to the constraint cost of migration. We design the ECP algorithm to solve the problem.

Keywords: Mobile edge computing · Mobile cloud gaming · Dynamic service placement · QoE impairment · User mobility

1 Introduction

With the development of 4G/5G technology and video games, mobile cloud gaming (MCG) has become the trend of the future gaming industry [1]. Mobile cloud gaming offers the possibility of running complex games on thin and light devices by offloading heavy tasks to the cloud, making mobile games accessible on any device and anywhere with simple settings. However, mobile edge computing often cannot fulfil increasingly stringent latency requirements because of the long-distance communication between the remote cloud and mobile devices [2].

As game services are devolved to edge networks, new challenges raised due to the limited resources of edge nodes and the unstable mobility of users. Existing works shows that network latency and frame rate are the major factors affecting the user's gaming

This work was supported in part by the National Natural Science Foundation of China under Grant 61662052, in part by the Natural Science Foundation of Inner Mongolia Autonomous Region under Grant 2021MS06002, in part by the Science and Technology Planning Project of Inner Mongolia Autonomous Region under Grant 2021GG0155, and in part by the Major Research Plan of Inner Mongolia Natural Science Foundation under Grant 2019ZD15.

experience [3]. Therefore, placing the service entity at a non-nearest but appropriate edge nodes is a more practical solution, which leads to the service placement problem.

In addition, users may move from one edge node service area to another while playing game. To maintain a satisfactory quality of service, service placement decisions should be continuously adjusted over time. An effective dynamic service placement strategy should: 1) coordinate communication and computation delays to minimize user-perceived delays; or 2) balance performance and cost in a cost-effective manner.

The main contributions of our work are summarized as follows:

- We propose a method to solve the problem of poor gaming experience in mobile cloud gaming through dynamic service placement in the MEC environment. Minimize the QoE impairment caused by low frame rate and long network delay.
- We design a dynamic service placement algorithm (ECP) under mobile cloud gaming. The algorithm transforms a long-term problem into a series of subproblems by applying the Lyapunov optimization framework.
- We conduct numerical simulations to evaluate the performance of ECP algorithm and measure the impact of the algorithm parameters. The results shows that the ECP algorithm outperforms other algorithms and has a large reduction on the long-term gaming experience impairment.

The rest of this paper is organized as follows. Section 2 reviews related work, the system model will be introduced in Sect. 3, and Sect. 4 proposes an online service placement algorithm to seek the optimal service placement strategy. Section 5 evaluates the performance of the algorithm and concludes the paper in Sect. 6.

2 Related Work

Our work is related to traditional mobile cloud gaming QoE issues as well as service placement problem in MEC systems. Research on the most relevant issues of these topics is reviewed in our article.

With the development of MEC, people begin to study the problem of service placement in MEC. A key challenge to achieve efficient service deployment in MEC is to track the mobility of users and devices. To meet these challenges, E.g., Nadembega [4] solved the trade-off between execution overhead and latency, using a mobility-based prediction scheme to estimate data transfer throughput, VM migration management, and switching time in advance. Also, Wang [5] looked at how to place services by predicting user location and preferences, database location and more. In [6] the authors assume a good understanding of the user's mobility within a given time frame. In this paper, our goal is to simultaneously optimize the service placement to minimize the QoE impairment caused by the network delay and the frame rate.

Existing MCG works can be divided into two categories: 1) Graphics streaming [7] where the data is the game specification model and instructions, and the rendering task is handed over to the mobile device. 2) Video streaming [8] where the data has been rendered as a game scene. Due to the thin-client and transparency feature, video streaming is the most used in practice. Therefore, the target of this article is also video

streaming. Video streaming faces two major challenges, namely, huge wide-area traffic and long network latency. However, since mobile devices still have to communicate with the remote cloud through the WAN, these works fail to reduce the traffic or the network delay to a desired grade.

3 System Model and Problem Formulation

As illustrated in Fig. 1, we consider a MEC system that consist of a set $\mathcal{N} = \{1, 2, \dots, N\}$ of edge nodes, a remote cloud and a network operator. Each edge node is a base station or a wireless access node equipped with a server, and consequently has the capability of computing and communication. Furthermore, edge nodes are connected via high-speed local-area network (LAN) and can communicate with remote clouds through the WAN. To facilitate capturing user mobility, it is assumed that services are provided in discrete slot frames, expressed as $\mathcal{T} = \{1, 2, \dots, T\}$. The system provides a game library $\mathcal{G} = \{1, 2, \dots, G\}$ for users to choose. And the user is represented as $\mathcal{M} = \{1, 2, \dots, M\}$. We denote $N_m^t \in \mathcal{N}$ as an edge node communicating with user m within time slot t .

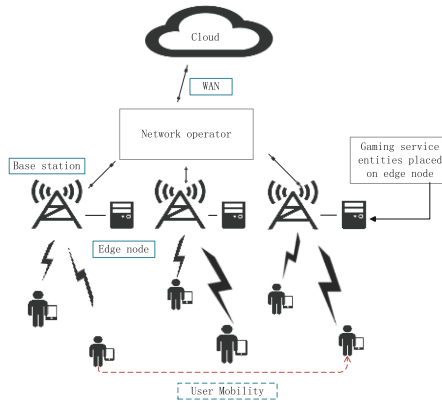


Fig. 1. Architecture of MEC-enable mobile cloud gaming

In order to maintain a satisfactory quality of service, here we take a binary indicator $x_m^t(n)$ to represent the dynamic service placement decision variable, let $x_m^t(n) = 1$ if the service entity of user $m \in \mathcal{M}$ is placed at the MEC node $n \in \mathcal{N}$ at slot t , and $x_m^t(n) = 0$ else. We have the following constraints on the service placement decision:

$$\sum_{n \in \mathcal{N}} x_m^t(n) = 1, \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{1}$$

$$x_m^t(n) \in \{0, 1\}, \forall m \in \mathcal{M}, \forall t \in \mathcal{T}, \forall n \in \mathcal{N} \tag{2}$$

Executing a service entity will cause the corresponding node to consume computing resources. We denote S_n as the computing power of node n and S_g as the computing

capacity (e.g., CPU cycles per second) required by the service entity running game g . We have the following constraint:

$$\sum_{m \in M} x_m^t(n) \times S_{gm} \leq S_n, \forall n \in N, \forall t \in T \tag{3}$$

Denoting the bandwidth capacity of edge node n at any time slot t as W_n^t , and denoting the bandwidth allocated to user m as y_m^t , analogous to the limitation of computing resources, we have the following constraints:

$$\sum_{m \in M, n_i^m = n} y_m^t \leq W_n^t, \forall n \in N, \forall t \in T \tag{4}$$

The frame rate is determined by the bandwidth and the resolution also with compression ratio. We assume that each game has only one resolution, and denote the resolution of game g by r_g . Thus, at any slot t , the allocation decision is y^t , and γ represents the transmission compression ratio the frame rate f_m^t of user can be expressed as:

$$f_m^t = \frac{y_m^t}{\gamma \times r_{gm}} \tag{5}$$

The communication delay includes network propagation delay and data transmission delay. Given the service request information and the current location of user m , the communication delay to MEC node n can be represented by the general model $H_n^m(t)$. When considering the service placement decision $x_m^t(n)$, the communication delay is $L^m(t) = \sum_{n=1}^N x_m^t(n)H_n^m(t)$.

In this paper, we use $R^m(t)$ to denote the amount of computation capacity required by service request of user m at time slot t . Computation delay in time slot t is $D^m(t) = \sum_{i=1}^M x_i^m(t)R^m(t)N_i(t)/F_i$ among them $N_i(t)$ is the number of users served by node i in time slot t and F_i represents the maximum computation power.

By combining the computation delay $D^m(t)$ and the communication delay $L^m(t)$, We denote the total delay that user m is in during period t as $T^m(t) = L^m(t) + D^m(t)$.

We use the mean opinion score (MOS) to evaluate the gaming experience, to obtain the video coding parameters of the QoE model. According to the collected data we fit the data by using different linear and nonlinear models, and the accuracy of the fitting is analyzed to obtain the following expression of the QoE impairment function:

$$I(t, m) = \alpha_1 f_m^t + \alpha_2 T^m t + \alpha_3 (T^m t)^2 + \alpha_4 (f_m^t)^2 + \alpha_5 f_m^t T^m t \tag{6}$$

The QoE impairment functions in terms of the network delay and the frame rate, and denote by $I(t, m)$. Among them $\alpha_1 - \alpha_5$ are model parameters of the game specification.

In this part, we present the migration cost. While dynamic service placement provides satisfactory QoE by migrating service entities between edges to follow user mobility. But it's worth noting that migrating across edge servers incurs additional operational costs. We use $C_{ij}^g(t)$ to denote the cost of migrating game task entity g from node i to node j in time slot t . Then at any time slot t , the service placement decisions x^t and x^{t-1}

are given, and the migration cost of user m is denoted as C_m^t , which can be expressed as follows:

$$C_m^t = \sum_{m=1}^M \sum_{i=1}^N \sum_{j=1}^N C_{ijm}^g(t) x_m^t x_m^{t-1} \quad (7)$$

Due to users mobility, to ensure the desired level of QoE, the service entity should be migrated to follow the user movement. We introduce C_{avg} to denote the long-term-averaged cost budget limit the cost of service migration within time slot t , which satisfies:

$$\lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=1}^T \sum_{m \in M} C_m^t \leq C_{avg} \quad (8)$$

Based on above formulation, our goal is to optimize service placement for MCG using MEC to minimize QoE impairment caused by frame rate and network latency. We describe the long-term time-averaged QoE impairment and migration cost as:

$$p1 : \min \lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=1}^T \sum_{m \in M} I(t, m) \quad (9)$$

$s.t.(1) - (4), (8)$

Striking the optimal solution requires a full knowledge of the system dynamics, such as user mobility patterns. Even if $p1$ is transformed into a series of one-shot optimization, the derived subproblem is a mixed integer non-linear optimization problem and is proved to be NP-hard in the following section.

4 Online Service Placement Algorithm

In this section, we describe a novel framework for making online service placement decisions. To solve $p1$, we transform the problem into a Lyapunov-based queue stability control problem, and present it in Algorithm 1.

Algorithm 1 The Placement Algorithm

Input: $q^0=0, \mathcal{N}, \mathcal{M}, \mathcal{T}, C_{avg}$;

Output: Service placement decisions $\{x_m^t(n) | t \in \mathcal{T}\}$;

1: $x_{m0}^0 \leftarrow 1, \forall m \in \mathcal{M}$;

2: **for** each time slot $t=1, 2, \dots, T$ **do**

3: Collect information $(n_m^t)_{m \in \mathcal{M}}, (p_{ij}^t)_{i, j \in \mathcal{N}}$;

4: Solve the problem $p2$ using **Algorithm 2**;

5: Update the virtual queue using Equation (11);

6: **end for**

In this part, we present the Lyapunov-based Online Algorithm. Due to the dynamic nature of the system, $p1$ makes a long-term trade-off between cost and performance without future global information. The key idea of Lyapunov optimization [9] is to transform the initial problem into a queue stability control problem by introducing a virtual queue for each long-term constraint. When the queue is controlled to be long-term stable, the corresponding long-term constraints are satisfied.

First, we construct the constraint cost as a dummy queue and assume that the initial queue backlog is 0 (i.e., $Q(0) = 0$):

$$Q(t + 1) = \max \left\{ Q^t + \sum_{m \in M} C_m^t - C_{avg}, 0 \right\} \tag{10}$$

where Q^t is the queue length of time slot t , which represents the cost of performing task migration at the end of the time slot.

We set the initial queue backlog to 0, i.e. $Q^t = 0$. We treat queue backlog as a criterion for evaluating violations of cost constraints. That is, when the value of Q^t is large, it means that the accumulated migration cost has exceeded the budget, and migration during this period should be avoided. The main part of our online service placement algorithm is to solve the following problem $p2$ in at every time slot t :

$$\min \quad V \times \sum_{m \in M} I_{t,m}(g, f, b) + Q^t \times \left(\sum_{m \in M} C_m^t - C_{avg} \right) \tag{11}$$

The positive parameter V is the Lyapunov control parameter, which is used to control the bias between the original objective and the long-term constraint. In order to solve the proposed problem, $p2$ needs to be solved.

Algorithm 2 The Algorithm ECP

- Input:** $(n_m^t)_{m \in M}, (p_{ij}^t)_{i,j \in N}, x^{t-1}, q^t, V;$
Output: Service placement decisions $\{x_m^t(n) | t \in T\};$
- 1: $x^t \leftarrow x^{t-1};$
 - 2: Calculate the objective value f under x^t
 - 3: **repeat**
 - 4: Randomly select a user $\hat{m} \in M$ and a node $\hat{n} \in N$
 - 5: and serve \hat{m} at \hat{n} and generate a new decision $\hat{x};$
 - 6: **if** \hat{x} is feasible **then**
 - 7: Calculate the objective value \hat{f} under $x^t;$
 - 8: With probability $\eta = \frac{1}{1+e^{(\hat{f}-f)/\beta}}$, adopt $x^t,$
 - 9: i.e., $x^t \leftarrow \hat{x}, f \leftarrow \hat{f};$
 - 10: keep x^t unchanged;
 - 11: **end if**
 - 12: **until** the underlay chain is stationary;
 - 13: return $x^t.$
-

We set the environment considered in $p2$ to consist of a remote cloud and a single edge node, and we assume that the bandwidth of the edge node is unlimited, so the QoE

impairment of frame rate can be ignored, assuming the queue backlog is zero, so, in this particular case, $p2$ can be rewritten as:

$$\max \sum_{m \in M} x_m^t \times I_{t,m}(d^t) \quad (12)$$

where d^t is the network delay between the edge node and the remote cloud, x_m^t is the service placement decision for user m and W is the computation power of the edge node. We can see that $p2$ becomes a standard 0 – 1 knapsack problem after rewriting, in which the knapsack capacity is W , the item set is M , and the value and weight of each set $m \in M$ are $I_{t,m}(d^t)$ and w_{gm} . Due to the NP-hardness of its special case, $p2$ is proved to be a NP-hard problem.

Since the NP problem cannot be solved in polynomial time when $P \neq NP$, we turn to the approximate optimal solution and show it as an iterative algorithm based on the Markov approximation method. Each iteration is constructed as a one-step transition from one state of the underlying chain to another state. The algorithm is shown in Algorithm 2, based on the first initial service placement decision x_m^t in the current state, $\hat{m} \in M$ and node $\hat{n} \in N$ are randomly selected in each iteration to generate a new service placement decision.

5 Performance Evaluation

In this section we perform numerical simulations to evaluate our ECP algorithm by comparing with several algorithms, in addition, we change the algorithm parameters and evaluate their impact on the algorithm performance.

In the simulation, like some previous work [10], we simulate a 2 km × 2 km area with 35 edge nodes. For simplicity, we assume that edge servers are irregular motion in this regular mesh network, and each user is within the coverage of exactly one edge node at any time. We use ONE to simulate user mobility, the user moves at a speed of 0.5 m/s to 1.5 m/s and a random direction will be chosen each time in the user's movement. The communication delay between cloud and edge nodes is within 200 ms to 300 ms. More parameters are shown in Table 1.

To evaluate the performance, following algorithms are used.

1. Stay at the cloud (SC) [11]: Mobile users always place service entities in the cloud, simulating traditional mobile cloud gaming.
2. Follow the user mobility (FUM) [11]: Regardless of the switching cost, and it's an ideal way, mobile users will always choose the nearest edge server to handle the request in each time slot.
3. First in first migration (FIFM): Migrate at minimal cost per user, repeat until budget is used up.
4. Based on Markov center algorithm (MCA) [5]: Service placement decisions are obtained using a Markov approximation method.

Figure 2 (a) shows the impact of cost budget on QoE impairments after a 200 slot simulation. Impairment values for SC and FUM are same because they are not affected

Table 1. Simulation parameter settings

Parameter	Value
User number M	600
Game number G	20
Computation capacities R^m	10 GHz–20 GHz
Bandwidth capacities W_n	20 MBps–50 MBps
Game resolutions r_g	{ 240 P, 360 P, 480 P, 720 P }
Migration cost budget C_{avg}	20
Lyapunov control parameter V	10^3
Compression ratio α	0.2
Communication requirement Q_n	0.5 GHz–1 GHz

by cost budgets. For FIFM, MCA and ECP, it can be observed that with the growth of cost budget, more services can follow the user’s mobility to reduce impairment.

Figure 2 (b) illustrates the QoE impairment per time slot. All user’s service entities are initially placed in the cloud and afterwards placed according to ECP and these compared alternatives. As expected, SC has highest QoE impairment due to its long-distance communications and FUM achieves the lowest conversely. FIFM repeatedly migrates the service with maximal migration cost until the cost budget is used up. By contrast, ECP achieves an elegant balance between the performance and the cost.

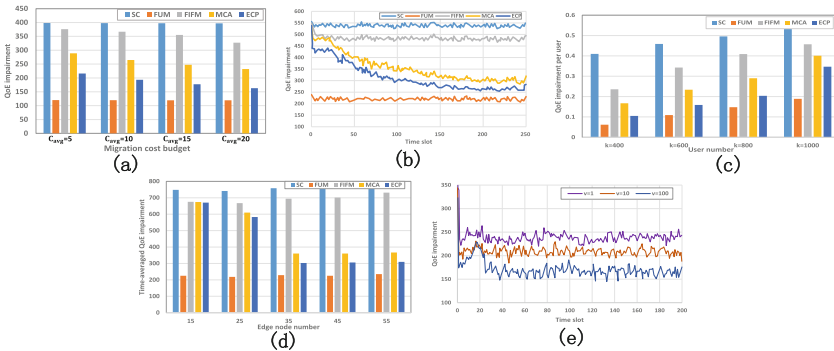


Fig. 2. Impact of different parameter on QoE impairment

In Fig. 2 (c), we vary the parameter number of users and compare the QoE impairment of each user between ECP and other algorithms. Due to the limited number of edge nodes, the larger the number of users, the more users are served by remote clouds, which leads to an increase in the average QoE impairment of users. Furthermore, as the number of users grows, the gap between ECP and other algorithms becomes larger. The impairment gap between the FUM and the ECP will also become larger.

We compare different number of edge nodes from 15 to 55 and show the QoE impairment results in Fig. 2 (d). All algorithms except FUM perform poorly when the number of nodes is small since most of the service entities are placed on the cloud. To some extent, increasing the number of nodes improve the performance. We learned that MCA performs poorer as the number increases from 35 to 55, this is because MCA prefers more to migrate services in the edge and ignores services placed in the cloud.

Figure 2 (e) shows the change of QoE impairment under different Lyapunov control parameters V . We learned that increase V from small values can improve the performance of ECP. We also observed that increase V at large values contributed little to performance and thus improving the performance by simply adjusting V is unviable.

6 Conclusion

In this paper, we investigate the dynamic service placement problem for MEC-enabled MCG and formulate a long-term optimization problem with random settings. We then propose ECP, which transforms the original problem into a series of residual optimization problems through the Lyapunov optimization framework. The performance of ECP is evaluated by numerical simulations, and the results show that ECP can significantly reduce the QoE impairment.

References

1. Wu, D., Ke, Y., He, J., Li, Y., Chen, M.: Mobile cloud gaming. In: Springer Encyclopedia of Computer Graphics and Games, pp. 1–7 (2017)
2. Choy, S., Wong, B., Simon, G., Rosenberg, C.: A hybrid edge-cloud architecture for reducing on-demand gaming latency. *Multimedia Syst.* **20**(5), 503–519 (2014). <https://doi.org/10.1007/s00530-014-0367-z>
3. Huang, C.-Y., Hsu, C.-H., Chen, D.-Y., Chen, K.-T.: Quantifying user satisfaction in mobile cloud games. In: ACM Proceedings of Workshop on Mobile Video Delivery, pp. 1–6 (2014)
4. A. Nadembega, A. S. Hafid and R. Brisebois: Mobility prediction model-based service migration procedure for follow me cloud to support QoS and QoE. In: Proceedings of the IEEE International Conference on Communications (ICC), pp. 1–6 (2016)
5. Ksentini, A., Taleb, T., Chen, M.: A markov decision process-based service migration procedure for follow me cloud. In: Proceedings of the IEEE International Conference on Communications (ICC) (2014)
6. Ceselli, A., Premoli, M., Secci, S.: Mobile edge cloud network design optimization. *IEEE/ACM Trans. Netw.* **25**(3), 1818–1831 (2017)
7. Liao, X., et al.: LiveRender: a cloud gaming system based on compressed graphics streaming. *IEEE/ACM Trans. Networking* **24**(4), 2128–2139 (2016)
8. Wang, S., Xu, J., Zhang, N., Liu, Y.: A survey on service migration in mobile edge computing. *IEEE Access* **6**, 23511–23528 (2018)
9. Neely, M.J.: Stochastic network optimization with application to communication and queueing systems. *Synth. Lect. Commun. Netw.* **3**(1), 1–211 (2010)
10. Ouyang, T., Li, R., Chen, X., Zhou, Z., Tang, X.: Adaptive user-managed service placement for mobile edge computing: an online learning approach. In: Proceedings of the IEEE Conference on Computer Communications, pp. 1468–1476 (2019)

11. Cao, T., Qian, Z., Wu, K., Zhou, M., Jin, Y.: Service placement and bandwidth allocation for MEC-enabled mobile cloud gaming. In: Proceedings of the 2021 IEEE 22nd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), pp. 179–188 (2021)



Cooperative Offloading Based on Online Auction for Mobile Edge Computing

Xiao Zheng¹, Syed Bilal Hussain Shah²(✉), Liqaa Nawaf³, Omer F. Rana⁴,
Yuanyuan Zhu⁵(✉), and Jianyuan Gan⁵

¹ School of Computer Science and Technology, Shandong University of Technology,
Zibo, China

² School of Computing and Mathematics Manchester Metropolitan University,
Manchester, UK

sshah@dah.edu.sa

³ Computer Science School of Technologies, Cardiff Metropolitan University,
Cardiff, UK

LLLNawaf@cardiffmet.ac.uk

⁴ School of Computer Science and Informatics, Cardiff University, Cardiff, UK

ranaof@cardiff.ac.uk

⁵ School of Software, Dalian University of Technology, Dalian 116620, China

{zhuyans, jianyuan_g}@mail.dlut.edu.cn

Abstract. In the field of edge computing, collaborative computing offloading, in which edge users offload tasks to adjacent mobile devices with rich resources in an opportunistic manner, provides a promising example to meet the requirements of low latency. However, most of the previous work has been based on the assumption that these mobile devices are willing to serve edge users, with no incentive strategy. In this paper, an online auction-based strategy is proposed, in which both users and mobile devices can interact dynamically with the system. The auction strategy proposed in this paper is based on an online approach to optimize the long-term utility of the system, such as start time, length and size, resource requirements, and evaluation valuation, without knowing the future. Experiments verify that the proposed online auction strategy achieves the expected attributes such as individual rationality, authenticity and computational ease of handling. In addition, the index of theoretical competitive ratio also indicates that the proposed online mechanism realizes near-offline optimal long-term utility performance.

Keywords: Online auction strategy · Collaborative computing offloading · Long-term utility

1 Introduction

With the continuous development of advanced wireless communication technology in recent years, the number of mobile devices has also exploded. First, these applications are typically resource-intensive, latency-sensitive, and computationally intensive. Second, the computing power required by mobile devices is still

severely limited by portability operations [1]. This presents a serious test for the future of mobile devices [2].

Offloading computing tasks is a fundamental solution to the problem of resource constraints [3]. Although cloud computing has made great achievements in the past many years, when users finally offload tasks to the public cloud, there is still the problem of long delay, especially in the environment of severe network congestion. In recent years, MEC has been designed as a promising computing paradigm for mobile services with ultra-low latency [4, 5]. Rather than offloading tasks to a remote cloud, mobile users address ultra-low latency issues by cooperating with end users or by performing computationally intensive tasks at the network edge of nearby facilities [6]. It is precisely because it is known that the MEC system performance can be effectively improved through the untapped resources of a large number of mobile devices, this paper studies the MEC framework of user cooperation [7]. Specifically, some mobile devices could partake these untapped resources to assist other edge users in offloading computing tasks [8].

The existing game is to guarantee the real reliability of the online strategy proposed in this paper. In addition to request-private conditions such as resource requirements and task evaluation involved in offline policies, this paper also needs to address new obstacles, namely ensuring the authenticity of start times and task durations [9]. False edge users purposefully use false reports of their private information to control market decision-making action to obtain high profits, which will deteriorate the long-term profit system [10, 11]. This paper adopts the technical expression social utility, which is determined as the total utility of edge end users and mobile devices. The crucial is to incentivize edge users to claim their realistic details through the right price. Only when users report false conditions will they get less utility than if they report true information. Its early classic work - the Vickrey-Clarke-Groves (VCG) strategy was used to develop a mechanism to prove its auction [12, 13]. However, the existing VCG algorithm is not suitable for the online situation, because its payment determination requires the optimal distribution results. In the case of uncertain future task requests, this paper cannot obtain those optimal solutions [14].

To address all the above problems and challenges, this paper develops an incentive mechanism for online auctions with the following properties: (1) The arrival and departure of computing tasks and mobile devices are dynamic at any time, and each task will be set up with a bundled resource package in the future; the auction decision-making behavior is matched between dynamic tasks and mobile devices. (2) The auction strategy designed in this paper is conducted in an online manner and does not do any suppositions about the arrival of future request information. Despite the premise that future information is not available, task assignment decision-making must be done instantaneously. The main contributions of the article are summarized as follows.

In this paper, an online incentive strategy is developed in a collaborative MEC environment for multi-type resource users. This paper deals with the generality of collaborative task execution: (1) Tasks are heterogeneous and need different amounts of diverse resources; (2) The number of tasks a mobile device can perform is limited by its resource capacity; (3) The performance of resource

supply and demand can affect its unit resource price. Therefore, two mechanisms are designed in this paper. One is an offline mechanism based on VCG, which is optimal as a benchmark. The other is a true online mechanism that only refers to the current request status to make decisions.

2 System Model

A. Mobile Edge Computing: Referring to here a MEC involving M edge users, indicated by $\mathcal{M} = \{1, 2, \dots, M\}$, microbase stations for mobile devices N , indicated by $\mathcal{N} = \{1, 2, \dots, N\}$ devices serving the users. Mobile devices can be thought of as smartphones, mobile microclouds, ipads and Internet of Things devices. It is assumed that the system is run in timeslot mode, and each timeslot is represented as $t \in \mathcal{T}$, $\mathcal{T} = \{1, 2, \dots, T\}$. User i 's j -th task is denoted as $\mathcal{T}_{ij} = \{t_{ij}, l_{ij}\}$, where t_{ij} stands for the start time of the task, l_{ij} stands for the length of the task, that is, the amount of timeslots used to accomplish the task. Therefore, the index number required to complete the time slot is expressed as $t'_{ij} = t_{ij} + l_{ij} - 1$. Z -Resources for instance CPU, RAM, and bandwidth are assumed. Define $a_{ij}^z(t)$ be the amount of z -resources required for slottime t , whose variable $a_{ij}^z(t)$ varieties with time, and its varieties will be different due to the heterogeneity of computing tasks. $A_{ij} = \{a_{ij}^1, a_{ij}^2, \dots, a_{ij}^Z\}$ is defined as a computing resource as a specified bundle, where $a_{ij}^z = \{a_{ij}^z(t) : \forall t \in [t_{ij}, t'_{ij}]\}$.

To give a mode for mobility, set t_n and s_n to the interval time and service duration of mobile devices $n \in N$ respectively. The computing resources of each mobile device are limited. Defining C_n^z indicates the maximum capacity of type z -type resources on mobile device N . Because the microbase station can access the all network state, it is a system controller that controls the decision making of task scheduling.

B. Auction Theory: In this paper, the interaction between edge users and mobile devices is modeled as an auction strategy, which edge users are regarded as bidders and mobile devices as sellers. The microbase-station is a trusted third-party auction manager who manages both parties and makes online decisions. Users on the edge ask nearby mobile devices to assist with tasks and provide some immediate reward when the task is completed. The stages of the auction process are as follows:

Set b_i^j to the bid of task \mathcal{T}_{ij} . The bidding prototype of the task \mathcal{T}_{ij} should denoted as $\sigma_i^j = \{t_{ij}, l_{ij}, A_{ij}, b_i^j\} \in \Sigma_i$, where Σ_i is the bidding group of edge user i . There are \mathcal{M}, \mathcal{N} and $\Sigma = \{\Sigma_1, \Sigma_2, \dots, \Sigma_M\}$, the auction manager can control a winning bid set \mathcal{W} and a task assignment scheme, i.e., to search a mapping: $\{\sigma_i^j : \sigma_i^j \in \mathcal{W}\} \rightarrow \{n : n \in \mathcal{N}\}$ and the payment of each winning bidder $\sigma_i^j \in \mathcal{W}$. Note here that each bid σ_i^j is private info for edge user i .

In a fake auction, the bidder will present the difference between his request and his actual request. For the purpose of distinguishing, the submitted bids are indicated by $\sigma_i^j = \{t_{ij}, l_{ij}, A_{ij}, b_i^j\}$, and the actual request info is indicated as $\bar{\sigma}_i^j = \{\bar{t}_{ij}, \bar{l}_{ij}, \bar{A}_{ij}, q_i^j\}$.

C. Offline Revenue Maximization Problem: The entire information about bidding and mobile devices is available in an offline environment. There is a tradeoff between the utility and the cost of completing a task, which in turn creates some utilities for the bidder. The bid assignment variable $y_n(\sigma_i^j)$ is given here, and $y_n(\sigma_i^j) = 1$ when the bid σ_i^j is assigned to the mobile device N . And the overall bid allocation strategy is $\mathcal{Y} = (y_n(\sigma_i^j) : \forall n \in \mathcal{N}, \forall \sigma_i^j \in \Sigma)$.

Bidding allocation strategy \mathcal{Y} is defined, $\Lambda = (\lambda_{ij})$ is a payment rule, and λ_{ij} indicates the payment of task \mathcal{T}_{ij} . In order to explore this tradeoff, this paper adopts welfare benefit maximization index, which is mainly characterized via system completion utility and mobile device service cost.

1. Computation Completion Utility: Set bid Σ and bid allocation strategy \mathcal{Y} , and the system utility that can be completed by computing the task will be expressed as:

$$U(\mathcal{Y}) = \sum_{\sigma_i^j \in \Sigma} \sum_{n \in \mathcal{N}} y_n(\sigma_i^j) \cdot b_i^j \tag{1}$$

2. Mobile Device Service Costs: The service cost of mobile devices mainly comes from its battery energy consumption. This paper applies a linear energy consumption mould according to resource consumption. It is understood that in the case of not using dynamic voltage frequency scaling, its energy consumption and CPU, RAM usage approximately show a linear relationship. Set $r_n^z(t)$ to represent the z -type resource usage on mobile device N at time t , and its relevant execution cost can be expressed as

$$E_n^z(r_n^z(t)) = \begin{cases} g_n^z r_n^z(t) & 0 \leq r_n^z(t) \leq C_n^z \\ +\infty & \text{otherwise} \end{cases} \tag{2}$$

which g_n^z indicates the energy consumption required to use unit z -type resource in each slottime on mobile device n .

The total resource consumption in \mathcal{T} is summarized by $\mathbf{r} = (r_n^z(t)) : \forall n \in \mathcal{N}, \forall z \in \mathcal{Z}, \forall t \in \mathcal{T}$. Therefore, its operating cost is:

$$\Omega_E(\mathbf{r}) = \sum_{n \in \mathcal{N}} \sum_{t \in \mathcal{T}} \sum_{z \in \mathcal{Z}} E_n^z(r_n^z(t)) \tag{3}$$

3. Utility Maximization Problem: Set Σ_{-i}^{-j} to the entire set of claimed bid profiles for all tasks except bid σ_i^j . That $(\sigma_i^j, \Sigma_{-i}^{-j})$ stand for the entire bidding situation. The user i 's utility function is: $\mu_{ij}(\sigma_i^j, \Sigma_{-i}^{-j}) = b_i^j - \lambda_{ij}(\sigma_i^j, \Sigma_{-i}^{-j})$, when exist $x_n(\sigma_i^j) = 1$. The total utility of a mobile device is to receive the total payment minus the cost of service. Social utility maximization problem (SUM) is the difference between the utility completed after task aggregation and the service cost. In short, the problem of maximizing social benefits in the system model in the article will be converted into the mixed integer programming problem as follows:

$$\max_{\mathcal{Y}, \mathbf{r}} SUM(\mathcal{Y}, \mathbf{r}) = U(\mathcal{Y}) - \Omega_E(\mathbf{r}) \tag{4}$$

$$\sum_{\sigma_i^j \in \Sigma: t_{ij} \leq t \leq t'_{ij}} y_n(\sigma_i^j) a_{ij}(t) \leq r_n^z(t) \quad \forall n, \forall t, \forall z \quad (4a)$$

$$\sum_{n \in \Psi_{ij}} y_n(\sigma_i^j) \leq 1 \quad \forall \sigma_i^j \quad (4b)$$

$$y_n(\sigma_i^j) \in \{0, 1\} \quad \forall \sigma_i^j \quad \forall n \in \Psi_{ij} \quad (4c)$$

which $\Psi_{ij} = \{n \in N : t_n \leq t_{ij}, l_{ij} \leq s_n\}$.

3 Offline Auction Strategy Formed

The objective of the article is to design a VCG enabled offline optimal auction strategy in which the auctioneer has the entire future details situation. The optimal allocation outline is the optimal solution of the precisely maximized mixed integer programming, namely, Eq. (4).

Strategy 1. (VCG-enabled Offline Auction Strategy-VCG-OOA)

- (1) The allocation strategy $\mathbf{Y}_n^O \triangleq (y_n^O(\sigma_i^j) \quad \forall \sigma_i^j \in \Sigma \quad \forall n \in \mathcal{N})$ is derived by optimal solution to the mixed integer programming problem with a union of global bid Σ .
- (2) The payment strategy $\mathbf{A}_n^O \triangleq (\lambda_n^O(\sigma_i^j) \quad \forall \sigma_i^j \in \Sigma \quad \forall n \in \mathcal{N})$, which $\lambda_n^O(\sigma_i^j)$ is described as :

$$\lambda_n^O(\sigma_i^j) = SUM(\mathcal{Y}^o(\Sigma), \mathbf{r}^o(\Sigma)) - b_i^j - SUM(\mathcal{Y}^o(\Sigma - \{\sigma_i^j\}), \mathbf{r}^o(\Sigma - \{\sigma_i^j\})) \quad (5)$$

where $\Sigma - \{\sigma_i^j\}$ indicates all bid sequences except bid b_i^j , and $\mathcal{Y}^o(\Sigma - \{\sigma_i^j\})$ indicates the optimal solution obtained when $\Sigma - \{\sigma_i^j\}$ treats as the input.

Algorithm 1. OAP-SUM Strategy

- 1: **Input:** Current Event;
 - 2: $\tilde{t} \leftarrow$ Now timeslot;
 - 3: $N(\tilde{t}) \leftarrow$ {n|the collection of mobile devices participating in the auction at \tilde{t} };
 - 4: $\Psi(\tilde{t}) \leftarrow$ { σ_i^j |bid has been authorized but work has not yet been processed within \tilde{t} }
 - 5: **if** Event==‘Mobile device n reaches’ **then**
 - 6: $N(\tilde{t}) \leftarrow N(\tilde{t}) \cup n, \quad (t_n \leq \tilde{t} \leq t'_n)$;
 - 7: **end if**
 - 8: **if** Event==‘Bid σ_i^j reaches’ **then**
 - 9: Computing the union Ψ_{ij} for bid σ_i^j based on $N(\tilde{t})$;
 - 10: $\mathcal{Y}(\sigma_i^j), \lambda_{ij} \leftarrow$ OAP-SUM-A ($\tilde{t}, N(\tilde{t}), \sigma_i^j, \Psi_{ij}$)
 - 11: $\Psi(\tilde{t}) \leftarrow \Psi(\tilde{t}) \cup \sigma_i^j$
 - 12: $\mathcal{Y} \leftarrow \mathcal{Y} \cup \mathcal{Y}(\sigma_i^j), A \leftarrow A \cup \lambda_{ij}$
 - 13: **end if return:** \mathcal{Y} and A
-

4 Online Auction Strategy Formation

OAP-SUM Strategy: According to the applicable rules of Myerson’s theorem, this paper mainly describes the scheme implementation and allocation rules of SUM online auction policy (OAP-SWM), as shown in Algorithm 1. This paper first develops an event processing application, which involves the call processing of results for instance bid arrival, bid acceptance, task completion, mobile device arrival and mobile device departure (line 2–4). OAP-SUM computes two collections, $N(\tilde{t})$ indicates the collection of mobile devices available at auction time \tilde{t} . $\Psi(\tilde{t})$ is the collection of accepted bids for unfinished tasks at \tilde{t} (line 5–6). The union $N(\tilde{t})$ is updated as soon as the new mobile device reaches. In the case of submitting a new bid, OAP-SUM first computes the union Ψ_{ij} of bid σ_i^j based on $N(\tilde{t})$ (line 7–9). OAP-SUM-A is controlled by function based on allocation decision and payment decision at lines 10. The OAP-SUM set is then updated at lines 11–12.

1) Allocation policy: This paper uses the primitive dual technique to develop allocation policy. Firstly, the relaxed integer constraint is adopted in Eq. (4c), and $y_n(\sigma_i^j) \in \{0, 1\}$ is converted to $y_n(\sigma_i^j) \geq 0$. Based on the standard Fenchel duality principle, the cost function or the conjugate function form $\hat{E}_n^z(x)$ of $E_n^z(r_n^z(t))$ is first given.

$$\hat{E}_n^z(x) = \max_{r_n^z(t) \geq 0} \{x r_n^z(t) - E_n^z(x)\} \tag{6}$$

The dual variables $\eta_n^z(t)$ and ν_{ij} are added to the Eqs. (4a) and (4b) in the form of constraint conditions. The duality problem is developed as follows:

$$\min \sum_{n \in \mathcal{N}} \sum_{z \in \mathcal{Z}} \sum_{t \in \mathcal{T}} \hat{E}_n^z(\eta_n^z(t)) + \sum_{\sigma_i^j \in \Sigma} \nu_{ij} \tag{7}$$

$$\nu_{ij} \geq b_i^j - \sum_{z \in \mathcal{Z}} \sum_{t_{ij} \leq t \leq t'_{ij}} \eta_n^z(t) a_{ij}^z(t) \quad \forall n, \forall \sigma_i^j \tag{7a}$$

$$\eta_n^z(t) \geq 0 \quad \forall n, \forall z, \forall t \tag{7b}$$

$$\nu_{ij} \geq 0 \quad \forall \sigma_i^j \tag{7c}$$

Based on the principle of complementary relaxation primal duality in the Karush -Kuhn -Tucker (KKT) condition, the primal variable $y_n(\sigma_i^j) = 1$ iff the dual constraint, i.e., Eq. (7a) is valid in the optimal solution. In order to realize the feasibility of the double restriction of formula (7a), for each new bid σ_i^j case, this paper defines:

$$\nu = [y]^+, \quad y = \max_{n \in \Psi_{ij}} (b_i^j - \sum_{z \in \mathcal{Z}} \sum_t \eta_n^z(t_{ij}, t) a_{ij}^z(t)) \tag{8}$$

which $[y]^+$ indicates $\max \{y, 0\}$. From this comes the allocation rule. In the case of $\nu_{ij} > 0$, bids σ_i^j will be accepted and $y_{n'}(\sigma_i^j) = 1$, otherwise, they will be rejected.

Algorithm 2. OAP-SUM-A ($\tilde{t}, N(\tilde{t}), \sigma_i^j, \Psi_{ij}$)

```

1: Initialize:  $\mathcal{Y}(\sigma_i^j) = (y_n(\sigma_i^j) \quad \forall n \in \mathcal{N}), \lambda_{ij} = 0;$ 
2: for  $\forall n \in \mathcal{N}(\tilde{t})$  do
3:   for  $\forall z \in \mathcal{Z}$ 
4:     for  $\forall t = \tilde{t} : T$  do
5:        $r_n^z(t, \tilde{t}) = \sum a_{ij}^z(t) \quad \forall \sigma_i^j \in \Psi(\tilde{t})$ 
6:        $\eta_n^z(t, \tilde{t}) = \Gamma_n^z(r_n^z(t, \tilde{t}))$ 
7:     end for
8:   end for
9: end for
10: Obtaining  $n'$  via resolving the Eq. (12);
11: Computing the dual variable value  $\nu_{ij}$ ;
12: Computing the union  $\Psi_{ij}$  for bid  $\sigma_i^j$  based on  $N(\tilde{t})$ ;
13:  $\nu_{ij} \leftarrow b_i^j - \sum_{z \in \mathcal{Z}} \sum_{t_{ij} \leq t \leq t'_{ij}} \eta_{n'}^z(t, \tilde{t}) a_{ij}^z(t)$ 
14:  $\Psi(\tilde{t}) \leftarrow \Psi(\tilde{t}) \cup \sigma_i^j$ 
15:  $\mathcal{Y} \leftarrow \mathcal{Y} \cup \mathcal{Y}(\sigma_i^j), \Lambda \leftarrow \Lambda \cup \lambda_{ij}$ 
16: if  $\nu_{ij} > 0$  then
17:    $y_{n'}(\sigma_i^j) \leftarrow 1$  and  $y_n(\sigma_i^j) \leftarrow 0 \quad \forall n \in \{\mathcal{N} - n'\}$ 
18:    $\lambda_{ij} \leftarrow \sum_{z \in \mathcal{Z}} \sum_{t_{ij} \leq t \leq t'_{ij}} \eta_{n'}^z(t, \tilde{t}) a_{ij}^z(t)$ 
19: else
20:    $\nu_{ij} \leftarrow 0$ 
21:    $y_n(\sigma_i^j) \leftarrow 0 \quad n \in \mathcal{N}$ 
22: end if
return:  $\mathcal{Y}(\sigma_i^j)$  and  $\lambda_{ij}$ 
    
```

(2) Payment Principle: The dual variable $\eta_n^z(t, \tilde{t})$ is taken as the optimal planned price for a z -type resource in time slot $t \geq \tilde{t}$ on mobile device n . Based on the off-line environment, the dual problem can be easily resolved directly to obtain these prices. However, it is difficult to get these prices when bids change dynamically over time. An auxiliary price function of $r_n^z(t) \in [0, C_n^z]$ is developed to realize online decision as soon as possible.

$$\Gamma_n^z = \frac{P_z - g_n^z}{2Z} \left(\frac{2Z(Q_z - g_n^z)}{P_z - g_n^z} \right)^{\frac{r_n^z(t)}{C_n^z}} + g_n^z \quad (9)$$

which $P_z = \min_{\sigma_{ij}} \frac{b_i^j}{\sum_{t \in [t_{ij}, t'_{ij}]} a_{ij}(t)}$, $Q_z = \max_{\sigma_{ij}} \frac{b_i^j}{\sum_{t \in [t_{ij}, t'_{ij}]} a_{ij}(t)}$ are respectively the

lower and upper limits of the bidder's valuation of unit z -type resources, which are obtained from previous information. All explanations are given in the future paper Appendix. In a word, the specific details of the allocation principle appear in OAP-SUM-A of Algorithm 2.

5 Experimental Analysis

A. Experimental Environment Setting: The task data in this paper is taken from Google database, which composed of task start time, implementation time

and resource requirement conditions. This article converts the request into a bid, as shown below. This paper assumes two types of resources, namely $Z = 2$. In order to obtain the bidding value of the task, the unit z -type resource valuation is taken from the random selection in P_z, Q_z , whose bidding value corresponds to its resource requirements within the quantization range of the unit valuation. The value range of P_z, Q_z varies with different experiments. $Q_z = 8$ and $P_z = 1$ are the default cases.

Under other default conditions, the edge computing system in this paper contains $N = 20$ mobile devices. In this paper, the cycle T of 300 timeslots is tried to run, and then the trajectory of the mobile device is randomly generated. It is supposed that mobile devices have the property of Poisson process arrival in average arrival interval N/T , and their service time interval s_n is selected uniformly and randomly in [15, 35], and the normalized resource capacity of each mobile device comes from the uniform and random distribution within the range of [0.25, 0.35]. g_n^z is uniformly and randomly distributed between the range [0.6 - 1].

B. Actual Acquisition of Competitive Ratio Analysis: This paper adopts online auction strategy to realize the comparison between the actual competition ratio and the corresponding theoretical one. The actual competition ratio is based on the OAP-SUM Algorithm to realize the ratio between the maximum actual social utility and the optimal offline social utility. The value of the theoretical competition ratio is $\ln(2Z\gamma)$, which $\gamma = \max_{n,z} \frac{Q_z - g_n^z}{P_z - g_n^z}$.

Figure 1(a) verifies the comparison between the actual and theoretical competition ratios of OAP-SUM Algorithm as the number of tasks grows. $Q_z = 8$, the paper learned that most of the actual competitive ratio is around 1.4, which is far less than the upper limit of the actual theoretical value, which denotes that the online strategy developed shows superior performance. However, the value of the actual competition ratio increased slightly with the grow in the number of tasks. The real reason is that with the grow of tasks, the future uncertainty will be more difficult to control, and the possibility of task allocation will be more and more, and the corresponding decision difficulty will be more and more uncontrollable. Furthermore, it is specifically understood that the theoretical ratio is not correlated with the corresponding number of tasks.

In Fig. 1(b), the functional forms of actual and theoretical competition ratios of OAP-SUM are studied when parameter γ ranges from 6 to 12. The number of tasks given here is 120 and the number of mobile devices is 20. Thus, with the increase of γ , the actual ratio will also increase. This makes sense because higher unit resource prices will lead to real improvements in competitiveness. The theoretical ratio is the same. This verification result is consistent with the analysis that the competition ratio is controlled by the value of $\ln(2Z\gamma)$.

C. Individual Rational Analysis: This paper studies the performance of OAP-SUM from the perspective of individual rationality, as illustrated in Fig. 2. Here, 20 successful bids are randomly selected from the winning collection, and the submitted bids, actual payments, and actual execution costs are given. It can be seen from Fig. 2 that the bid submitted is continuously greater than the actual

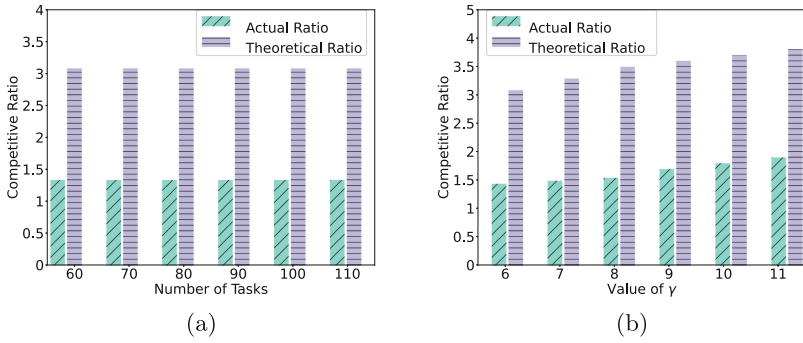


Fig. 1. Competitive Ratio of different number of tasks with γ .

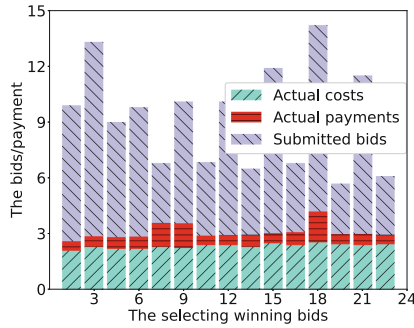


Fig. 2. Individual rational analysis

payment price paid to mobile devices, i.e., the individual rationality is guaranteed by OAP-SUM. **D. True Validity Analysis:** Now we study the analysis of the true validity of OAP-SUM. Figure 3 shows the performance impact of unreal resource requirements and task execution time on user utility respectively. In this paper, a winning bid σ_i^j is randomly selected and its bid situation is adjusted at any time. Meanwhile, the OAP-SUM algorithm is run again with other bids unchanged. It is stated here that the user cannot declare that the execution time is shorter than the actual execution time and the actual resource demand is less. Therefore, this article only applies to the environment where the user claims that the execution time is longer and the resource demand is higher. The value on the x -axis refers to the ratio of claimed requirements to actual resource requirements. It follows from this that submitting more bids than actual resource requirements will reduce the user's utility, while the actual resulting true resource requirements will yield the highest utility.

Similarly, the added task execution time in Fig. 3(b) is a bid σ_i^j in the range from 10 to 20, and its real value is 10. As can be seen from the graph, submitting bids with longer execution times than actual times will reduce the user's utility, while actual verified true execution times will yield the highest utility.

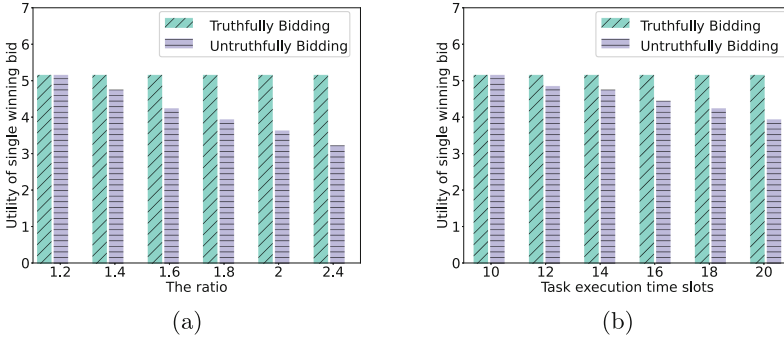


Fig. 3. Authenticity analysis

E. Comparison of the Two Proposed Strategies: The online mechanism OAP-SUM is now compared with the offline one VCG-OOA according to two capability indexes of user utility and winner percentage.

Figure 4 verifies the function comparisons of VCG-OOA and OAP-SUM in the light of user utility and winner percentage, respectively, as the number of tasks grows. It can be concluded that the user utility of OAP-SUM is larger or smaller than that of VCG-OOA in Fig. 4(a). The reason for this is that although there is an allocation optimum, VCG-OOA may not necessarily be the payment optimum. Figure 4(b) verifies the difference in function comparison of winner percentages. It is worth mentioning that the percentage of winners serves as a measure of distribution efficiency. Typically, the allocation strategy of OAP-SUM is close to optimal because its winner percentage is very close to the result of VCG-OOA. It can also be seen from the figure that as the amount of tasks grows, the percentage of winners will decrease. The reason behind this is that owing to the restricted resource capacity of mobile devices, the increased task value is greater than the increased number of winners.

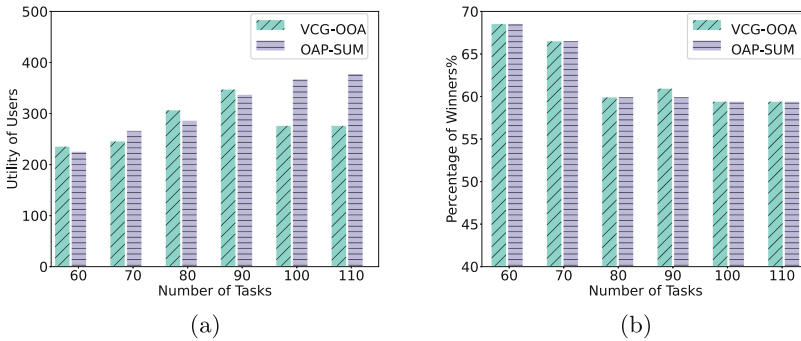


Fig. 4. Analysis and comparison of different number of tasks

Figure 5 verifies how the winner's utility and the winner's percentage are influenced when the ratio γ is increased from 6 to 12. Figure 5(a) shows that the utility of VCG-OOA is to maintain a kind of stability, while the utility of OAP-SUM decreases with the grow of γ value. The major reason is that the ratio σ is only a range of the marginal price function of the OAP-SUM strategy and will not affect the VCG-OOA strategy. OAP-SUM takes advantage of the increasing value of γ , which in turn charges the winner more to reduce user utility. Similarly, as displayed in Fig. 5(b), the percentage of winners shows a decreasing trend with the increase of γ value.

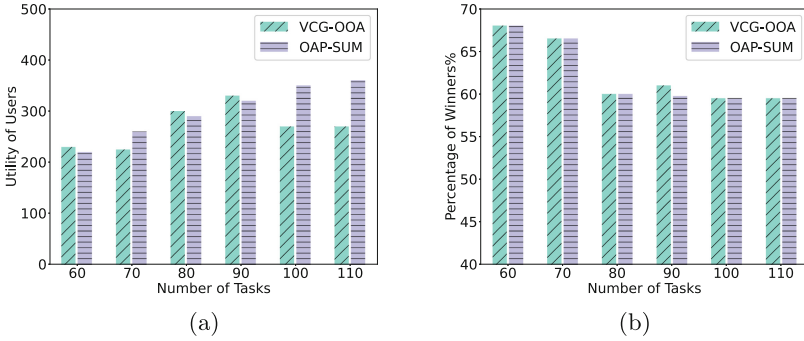


Fig. 5. Comparative analysis of different γ

6 Conclusion

In this paper, cooperative computing offload performance in MEC is investigated. In this paper, task offloading scheduling is modeled as an NP-hard SUM problem, and an offline optimization strategy is first used as a reference benchmark. This paper further designs an online strategy that does not rely on future details, which not only schedules computing tasks and computing payments in polynomial time without involving future details, but also optimizes the long-term social utility problem in a near-optimal fashion. A large number of theoretical analyses show that the designed online auction achieves such properties as individual rationality, authenticity and computational tractability. Meanwhile, function evaluations on actual world trajectories also validate the valid performance of the online mechanism proposed in this paper.

References

1. Newman, M.: Mobile Trends 2020: mobile trends for the next 10 - a collaborative outlook (2013)
2. Jošilo, S., Dán, G.: Selfish decentralized computation offloading for mobile cloud computing in dense wireless networks. *IEEE Trans. Mob. Comput.* **18**, 207–220 (2018)

3. Xu, D., Yong, L., Chen, X., et al.: A survey of opportunistic offloading. *IEEE Commun. Surveys Tutorials* **20**(3), 2198–2236 (2018)
4. Zhang, S., Zhang, N., Zhou, S., et al.: Energy-aware traffic offloading for green heterogeneous networks. *IEEE J. Sel. Areas Commun.* **34**(5), 1116–1129 (2016)
5. Ju, R., Hui, G., Xu, C., et al.: Serving at the edge: a scalable IoT architecture based on transparent computing. *IEEE Network* **PP**(5), 12–21 (2017)
6. Liu, G., Wang, J., Tian, Y., et al.: Mobility-aware dynamic service placement for edge computing. *EAI Endorsed Trans. Internet Things* **5**(19), 163922 (2018)
7. Zhou, Y., Zhang, D., Xiong, N., et al.: Post-cloud computing paradigms: a survey and comparison. *Tsinghua Sci. Technol.* **22**, 714–732 (2017)
8. He, J., Di, Z., Zhou, Y., et al.: Towards a truthful online auction for cooperative mobile task execution. In: 2018 IEEE Smart World, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCOM/IOP/SCI). IEEE (2018)
9. Tang, W., Zhang, K., Ren, J., et al.: Flexible and efficient authenticated key agreement scheme for BANs based on physiological features. *IEEE Trans. Mob. Comput.* **18**, 845–856 (2018)
10. Tang, W., Ren, J., et al.: Enabling trusted and privacy-preserving healthcare services in social media health networks. *IEEE Trans. Multimedia* **21**, 579–590 (2018)
11. Xu, C., Ren, J., Zhang, D., et al.: GANobfuscator: mitigating information leakage under GAN via differential privacy. *IEEE Trans. Inf. Forensics Secur.* **PP**(9), 1 (2019)
12. Wang, Y., Cai, Z., Tong, X., et al.: Truthful incentive mechanism with location privacy-preserving for mobile crowdsourcing systems. *Comput. Netw.* **135**, 32–43 (2018)
13. Wang, X., Chen, X., Wu, W.: Towards truthful auction mechanisms for task assignment in mobile device clouds. In: IEEE INFOCOM 2017 - IEEE Conference on Computer Communications. IEEE (2017)
14. Wu, B., Chen, X., Jiao, L.: Collaborative computing based on truthful online auction mechanism in internet of things. In: International Conference on Collaborative Computing: Networking, Applications and Worksharing. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-92638-0_9



Incentive Offloading with Communication and Computation Capacity Concerns for Vehicle Edge Computing

Chenliu Song, Ying Li^(✉), Jianbo Li, and Chunxin Lin

College of Computer Science and Technology, Qingdao University, Qingdao, China
yingli2016@qdu.edu.cn

Abstract. With the popularity of intelligent vehicles, computation-intensive vehicle tasks rise dramatically. Vehicle edge computing (VEC) is a promising technology that offloads overloaded computation tasks of intelligent vehicles to the edge. However, VEC servers are constrained by their available computation capacity while dealing with numerous tasks. To this end, we propose multi-party cooperation to complete vehicle task offloading. Computation-assisted vehicles (CAVs) with free resources assist VEC servers to offload Computation-required vehicles (CRVs), which enables computation resources of VEC servers and CAVs for CRVs' task execution. To motivate positive participation of VEC servers and CAVs, we design a resource management and pricing mechanism by quantifying their gains and costs. Such design efficiently integrate and leverage the communication mode and computing mode among participants to describe their interactions, which composes two two-stage Stackelberg games. While Nash equilibrium (NE) for each Stackelberg game reaches, none of participants violates unilaterally. Simulation results demonstrate its effectiveness of the proposed model.

Keywords: Vehicle edge computing · Incentive mechanism · Stackelberg game · Resource allocation

1 Introduction

With the rapid development of Internet of Things (IoT), various transport system applications win growing popularity in recent years. Intelligent vehicle services generate high performance demands, such as low-latency communication and intensive computation. Vehicle edge computing (VEC) thus emerges for

Supported by National Natural Science Foundation of China under Grant No. 61802216, National Key Research and Development Plan Key Special Projects under Grant No. 2018YFB2100303, Shandong Province colleges and universities youth innovation technology plan innovation team project under Grant No. 2020KJN011, Program for Innovative Postdoctoral Talents in Shandong Province under Grant No. 40618030001.

supporting computing, storage and network bandwidth [1]. Many studies introduce a variety of VEC technology, including vehicle fog computing [2], hybrid combination of fog computing architecture and vehicle cloud computing [3] and so on. VEC servers execute computation tasks, generated by the vehicles at the end, while reducing the delay. To improve the QoS of vehicle applications, studies have investigated if peripheral vehicles with free resources cooperate with VEC server to complete computing tasks. Authors in [4] studied task scheduling scheme for enhancing computing resource utilization with system stability and low latency. Some other studies on vehicle edge computing are also discussed in [5, 6].

In this paper, we are motivated to achieve mutual cooperation for vehicle task offloading. Vehicles, constrained by limited resources, require VEC serves for task execution. A large number of computation tasks burden VEC servers. To improve VEC efficiency, we consider numerous vehicles with free resources to assist the computing work of VEC servers. However, due to energy usage, most vehicles unenthusiastically participate in task calculation. In doing so, a framework of resource allocation and pricing is designed to economically compensate their costs and stimulate their cooperation.

2 System Module and Game Formulation

2.1 System Model

Our model focuses on task offloading between one VEC server and a group of vehicles. The proposed model is illustrated. Assume that a VEC server connects m CRVs and n CAVs within its coverage, denoted by $\mathcal{M} = \{1, 2, \dots, m\}$ and $\mathcal{N} = \{1, 2, \dots, n\}$. We also assume CRV i 's total computation tasks as w_i^r , $i \in \mathcal{M}$. Let \mathbf{p}^r denote CRVs' transmit powers over the VEC server's sub-channels with $\mathbf{p}^r = \{p_1^r, p_2^r, \dots, p_m^r\}$ while offloading tasks to VEC server. Specially, p_i^r is the transmit power from CRV i to VEC server, $i \in \mathcal{M}$. Similarly, the allocated powers for transmitting tasks to CAVs are denoted as $\mathbf{p}^a = \{p_1^a, p_2^a, \dots, p_n^a\}$, where p_j^a corresponds to the transmit power from VEC server to CAV j , $j \in \mathcal{N}$.

According to the communication model in [7], the channel rate from CRV i , $i \in \mathcal{M}$ to VEC server is defined,

$$R_i^r = B_i^r \log_2 \left(1 + \frac{p_i^r h_i^r}{\sigma_i^2} \right), \quad (1)$$

The channel rate from VEC server to CAV j is also defined,

$$R_j^a = B_j^a \log_2 \left(1 + \frac{p_j^a h_j^a}{\sigma_j^2} \right), \quad (2)$$

where B_i^r and B_j^a represent the transmission bandwidth of the subchannel, σ_i and σ_j represent noise power, and h_i^r and h_j^a represent the subchannel power gain.

We here define the utility function of a CRV as the difference between its gains and its costs. Service satisfaction contributes to its gain. Local computation cost, incentive payment and task transmission cost incur its total costs. Let T_i^r represent the uplink transmission time of delivering CRV i 's tasks to VEC server, which indicates its task size of $R_i^r T_i^r$. Given the unit pricing u_0 that VEC server charges CRVs, we define the utility of CRV $i, i \in \mathcal{M}$,

$$U_i^r(u_0, p_i^r) = s_i^r R_i^r T_i^r - \varepsilon_i^r (w_i^r - R_i^r T_i^r) - u_0 p_i^r - \xi_i^r p_i^r, \quad (3)$$

where s_i^r is CRV i 's internal demand rate, ε_i^r is the unitary local computation cost and ξ_i^r is the unitary transmission cost. Accordingly, the utility of CAV $j, j \in \mathcal{N}$ is outlined as,

$$U_j^a(u_j^a, p_j^a) = u_j^a p_j^a - \varepsilon_j^a R_j^a T_j^a, \quad (4)$$

where u_j^a is the incentive gain per unitary transmit power, ε_j^a is the unitary computation consumption. The parameter T_j^a represents computation tasks' uplink transmission time while delivering from VEC server to CAV j . It indicates the task size of $R_j^a T_j^a$, offloaded by CAV j . For the VEC server, its gain is the payment offered by CRVs. The total costs include local computation cost, incentive payment for CAVs, and power cost of transmitting task to the CAVs. So, the utility function U^{vec} of VEC server can be defined as:

$$U^{vec}(u_0, \mathbf{u}^a, \mathbf{p}^r, \mathbf{p}^a) = \sum_{i=1}^{\mathcal{M}} u_0 p_i^r - \sum_{j=1}^{\mathcal{N}} u_j^a p_j^a - \tau \sum_{j=1}^{\mathcal{N}} p_j^a - \varepsilon_0 \left(\sum_{i=1}^{\mathcal{M}} R_i^r T_i^r - \sum_{j=1}^{\mathcal{N}} R_j^a T_j^a \right), \quad (5)$$

where ε_0 is the unitary computation cost of VEC server and τ is the unit transmission cost. Denote \mathbf{u}^a as $\mathbf{u}^a = \{u_1^a, u_2^a, \dots, u_n^a\}$. For ease of illustration, we decompose the utility function in (5) into $U^{vec} = U_1^{vec} + U_2^{vec}$, in which,

$$U_1^{vec}(u_0, \mathbf{p}^r) = \sum_{i=1}^{\mathcal{M}} u_0 p_i^r - \varepsilon_0 \sum_{i=1}^{\mathcal{M}} R_i^r T_i^r, \quad (6)$$

$$U_2^{vec}(\mathbf{u}^a, \mathbf{p}^a) = \sum_{j=1}^{\mathcal{N}} \varepsilon_0 R_j^a T_j^a - \sum_{j=1}^{\mathcal{N}} (u_j^a + \tau) p_j^a. \quad (7)$$

2.2 Problem Formulation

In the three-tiered model, each participant has its own strategy and goal. Their different goals formulate different computation task management and pricing problems.

For player CRV i , $\forall i \in \mathcal{M}$, its strategy is to determine the transmit power p_i^r from it to VEC server in order to maximize the utility in (3). So, the CRV i 's optimization problem is stated as,

$$p_i^{r*} = \arg \max_{p_i^r > 0} U_i^r(u_0, p_i^r) \quad (8a)$$

$$\text{subject to } \underline{p}^r \leq p_i^r \leq \bar{p}^r, \quad (8b)$$

where \underline{p}^r and \bar{p}^r correspond to minimal and maximum transmit powers of CRVs.

For the player of VEC server, its strategy is to decide the pricing u_0 of CRVs's computation tasks and transmit power \mathbf{p}^a from it to CAVs in order to maximize the utility in (5). So, the VEC server's optimization problem is stated as,

$$(\mathbf{p}^{a*}, u_0) = \arg \max_{(p_j^{a*}, u_0)} U^{vec}(u_0, \mathbf{u}^a, \mathbf{p}^r, \mathbf{p}^a) \quad (9a)$$

$$\text{subject to } \underline{p}^r \leq p_i^r \leq \bar{p}^r, \quad (9b)$$

$$\underline{p}^a \leq p_j^a \leq \bar{p}^a, \quad (9c)$$

where \underline{p}^a and \bar{p}^a correspond to minimal and maximum transmit powers of CAVs.

Since the utility function in (5) can be separately divided into $U^{vec} = U_1^{vec} + U_2^{vec}$, problem (9) is thus decomposed into two subproblems as follows,

$$u_0^* = \arg \max_{u_0} U_1^{vec}(u_0, \mathbf{p}^r) \quad (10a)$$

$$\text{subject to } \underline{p}^r \leq p_i^r \leq \bar{p}^r, \quad (10b)$$

and,

$$p_j^a = \arg \max_{p_j^a} U_2^{vec}(u_j^a, p_j^a) \quad (11a)$$

$$\text{subject to } \underline{p}^a \leq p_j^a \leq \bar{p}^a. \quad (11b)$$

For player CAV j , $\forall j \in \mathcal{N}$, its strategy is to determine the pricing u_j^a of VEC server's overloaded tasks for maximizing the utility in (4). So, the CAV j 's optimization problem is described as,

$$u_j^{a*} = \arg \max_{u_j^a > 0} U_j^a(u_j^a, p_j^a) \quad (12a)$$

$$\text{subject to } \underline{p}^a \leq p_j^a \leq \bar{p}^a. \quad (12b)$$

2.3 Stackelberg Game Building

We here build two consecutive Stackelberg games for describing the interactions among CRVs, CAVs and VEC server, shown in Fig. 2. The VEC server acts as the leader of CRVs to determine the power pricing u_0 for CRVs' task. CRVs, as the followers of VEC server, then decide their transmit power strategy \mathbf{p}^r . The interactions between CRVs and VEC server formulates the first two-stage Stackelberg game. Problem (8) and Problem (10) jointly formulates such noncooperative game $\mathcal{G}^{CRV} = \{\mathcal{M}, \{p_i^r\}_{i \in \mathcal{M}}, \{U_i^r\}_{i \in \mathcal{M}}\}$, where \mathcal{M} is the set of CRVs, $\{p_i^r\}_{i \in \mathcal{M}}$ is the strategy set and $\{U_i^r\}_{i \in \mathcal{M}}$ is the cost function set.

CAVs, as the leaders, determine power pricing \mathbf{u}^a for VEC server's overloaded tasks. Accordingly, VEC server acts as the follower of CAVs in the second stage to determine its transmit power strategy \mathbf{p}^a . The interaction between CAVs and VEC server formulates the second two-stage Stackelberg game. In detail, Problem (11) and Problem (12) composes CAVs' game $\mathcal{G}^{CAV} = \{\mathcal{N}, \{u_j^a\}_{j \in \mathcal{N}}, \{U_j^a\}_{j \in \mathcal{N}}\}$, where \mathcal{N} is the set of CAVs, $\{u_j^a\}_{j \in \mathcal{N}}$ is the strategy set and $\{U_j^a\}_{j \in \mathcal{N}}$ is the set of profit functions.

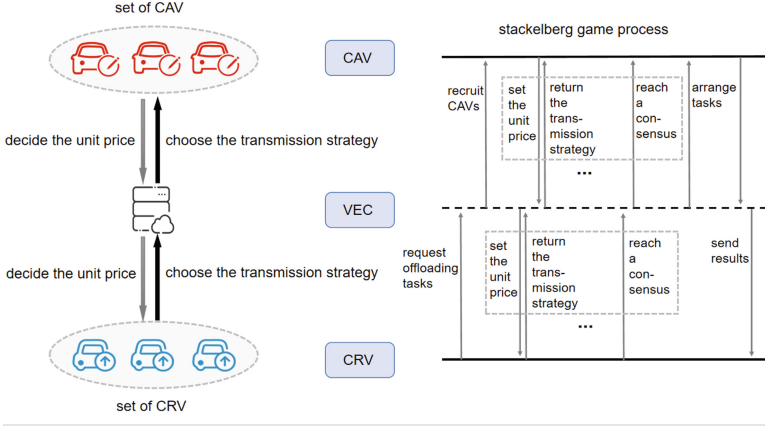


Fig. 1. Interactions in Stackelberg game

Definition 1. *Subgame Perfect Equilibrium (SPE) in \mathcal{G}^{CRV} :* If the stratification strategy $\{u_0^*, \mathbf{p}^{r*}\}$ represents SPE, it can reach NE in each stage of subgame, i.e.:

$$\begin{cases} \text{Stage I: } u_0^* = \arg \max_{u_0} U_1^{vec}(u_0, \mathbf{p}^r) \\ \text{Stage II: } \mathbf{p}_i^{r*} = \arg \max_{\mathbf{p}_i^r} U_i^r(u_0, \mathbf{p}_i^r) \forall i \in \mathcal{M} \end{cases} \quad (13)$$

Definition 2. *Subgame Perfect Equilibrium (SPE) in \mathcal{G}^{CAV} :* If the stratification strategy $\{\mathbf{u}^{a*}, \mathbf{p}^{a*}\}$ represents SPE, it can reach the NE in each stage of subgame, i.e.:

$$\begin{cases} \text{Stage I: } u_j^{a*} = \arg \max_{u_j^a} U_j^a(u_j^a, \mathbf{p}_j^a), \forall j \in \mathcal{N} \\ \text{Stage II: } \mathbf{p}^{a*} = \arg \max_{\mathbf{p}^a} U_2^{vec}(u_j^a, \mathbf{p}^{a*}) \end{cases} \quad (14)$$

Each game reaches SPE, which corresponds to a stable solution point of these problems. Under NE, no player has incentive to deviate. In detail, we use backward induction to analyze Stackelberg game. CRVs initiate the transmission strategy \mathbf{p}^r based on VEC server's pricing u_0 . To finish CRVs' computation tasks, the VEC server also recruits computation-assisted vehicles. CAVs determine a power pricing for VEC server's overloaded tasks, on which VEC server

allocates the transmit power \mathbf{p}^a for CAVs. The process continues until Nash equilibrium reaches. The best strategy $\{u_0^*, \mathbf{u}^a, \mathbf{p}^r, \mathbf{p}^a\}$ corresponds to the Nash equilibrium point, which is solved in the following theorems.

Theorem 1. *Given the unit price u_0 , the optimal strategy \mathbf{p}^r of CRVs is calculated in (15).*

$$p_i^{r*} = \gamma_i^r(\mathbf{p}^r) = \begin{cases} \underline{p}^r & \text{if } \frac{(s_i^r + \varepsilon_i^r)B_i^r T_i^r}{\ln 2(u_0 + \xi_i^r)} - \frac{\sigma_i^2}{h_i^r} < \underline{p}^r, \\ \frac{(s_i^r + \varepsilon_i^r)B_i^r T_i^r}{\ln 2(u_0 + \xi_i^r)} - \frac{\sigma_i^2}{h_i^r} & \text{otherwise,} \\ \bar{p}^r & \text{if } \frac{(s_i^r + \varepsilon_i^r)B_i^r T_i^r}{\ln 2(u_0 + \xi_i^r)} - \frac{\sigma_i^2}{h_i^r} > \bar{p}^r. \end{cases} \quad (15)$$

Theorem 2. *VEC server achieves the unique optimal pricing strategy u_0^* by solving subproblem (10) while $p_i^r \in [\underline{p}^r, \bar{p}^r], i \in \mathcal{M}$.*

Proof. Substituting (15) into (6), we calculate the first derivative and second derivative of U_1^{vec} with respect to u_0 ,

$$\frac{\partial U_1^{vec}}{\partial u_0} = \sum_{i=1}^M \frac{(s_i^r + \varepsilon_i^r + \varepsilon_0) B_i^r T_i^r}{\ln 2(u_0 + \xi_i^r)} - \frac{\sigma_i^2}{h_i^r} - \frac{u_0 (s_i^r + \varepsilon_i^r) B_i^r T_i^r}{\ln 2(u_0 + \xi_i^r)^2}, \quad (16)$$

$$\frac{\partial^2 U_1^{vec}}{\partial u_0^2} = -\sum_{i=1}^M B_i^r T_i^r \left[\frac{2(s_i^r + \varepsilon_i^r) \xi_i^r + \varepsilon_0 (u_0 + \xi_i^r)}{\ln 2(u_0 + \xi_i^r)^3} \right]. \quad (17)$$

Obviously, the second derivative of U_1^{vec} on u_0 is always negative. So, U_1^{vec} is a strict concave function with respect to u_0 , which indicates the uniqueness of optimal solution u_0^* . Because of non-linear complexity of $\frac{\partial U_1^{vec}}{\partial u_0}$ in (16), we approximate u_0^* by gradient ascent method.

Theorem 3. *VEC server achieves the unique optimal transmit power strategy \mathbf{p}^{a*} in (18) by solving subproblem (11) while $p_j^a \in [\underline{p}^a, \bar{p}^a], j \in \mathcal{N}$.*

$$p_j^{a*} = \delta_j^a(\mathbf{p}^a) = \begin{cases} \underline{p}^a & \text{if } \frac{\varepsilon_0 T_j^a B_j^a}{\ln 2(u_j^a + \tau)} - \frac{\sigma_j^2}{h_j^a} < \underline{p}^a, \\ \frac{\varepsilon_0 T_j^a B_j^a}{\ln 2(u_j^a + \tau)} - \frac{\sigma_j^2}{h_j^a} & \text{otherwise,} \\ \bar{p}^a & \text{if } \frac{\varepsilon_0 T_j^a B_j^a}{\ln 2(u_j^a + \tau)} - \frac{\sigma_j^2}{h_j^a} > \bar{p}^a. \end{cases} \quad (18)$$

Theorem 4. *CAV $j, j \in \mathcal{N}$ reaches the optimal pricing u_j^{a*} in (19) by solving problem (12),*

$$u_j^{a*} = R_j^a(\mathbf{u}^a) = \frac{\sqrt{(\Delta_j^1 - \Delta_j^2 - 2\Delta_j^3 \tau)^2 + 4\tau \Delta_j^3 (\Delta_j^1 - \Delta_j^3 \tau)} - (2\Delta_j^3 + \Delta_j^2 - \Delta_j^1)}{2\Delta_j^3}, j \in \mathcal{N}, \quad (19)$$

with $\Delta_j^1 = (\varepsilon_0 + \varepsilon_j^a) T_j^a B_j^a, \Delta_j^2 = T_j^s B_j^a \varepsilon_0$ and $\Delta_j^3 = \ln 2 \frac{\sigma_j^2}{h_j^a}$.

3 Performance Evaluation

3.1 Simulation Setup

We consider a simulation scenario in which a VEC server provides services for m CRVs with assistance of n CAVs for task offloading. Each VEC server is able to cover about 10–30 CRVs. Referring to [8–10], the default model parameters are stated as follows. It is assumed that the bandwidth of each subchannel is 20 MHz and its channel gain is 53 dbm. The noise power of the channel of the relevant vehicles is 10 db. The task transmission time T_i^r and T_j^a is set from 0.5 ms to 1.5 ms. CRVs’ satisfaction on offloaded tasks follows a normal distribution $N(e_0, \sigma_0)$. The number of CRVs and CAVs are 10 and 10 by default.

3.2 Simulation Results

We first investigate the effect of VEC server’s computation cost on its utility under different number of participants. We see from Fig. 2(a) and (b) that the increasing number of CRVs and CAVs is indeed helpful for enhancing VEC utility. While fixing the CRV number and CAV number, we start with observing a descending trend of VEC utility on computation cost. However, VEC server’s utility finally improves with incremental computation cost. Such situation is illustrated in Fig. 2(c) and (d). The growing computation cost incurs CRV’s decreasing contribution on VEC utility and CAV’s increasing contribution on VEC utility. While VEC server’s payoff increased by CRV completely cover that decreased by CAV, VEC utility turns into improvement.

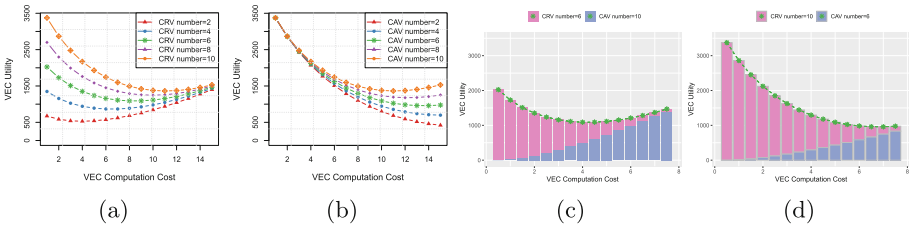


Fig. 2. Effect of computation cost on utility of VEC server

We next observe the comparison by altering the transmit power cost of VEC server. Figure 3(a) and (b) shows an ascending trend of VEC utility while more CRVs and CAVs participate in computation offloading. We also observe that the utility of VEC server tends to decline with the increasing VEC power cost in Fig. 3(a) and (b). Such result is explained in Fig. 3(c) and (d). VEC power cost has a negative effect on CAV’ contribution for VEC server’s payoff. However, CRV’s contribution on VEC server’s payoff don’t follow a significant change. Accordingly the utility of VEC server decreases.

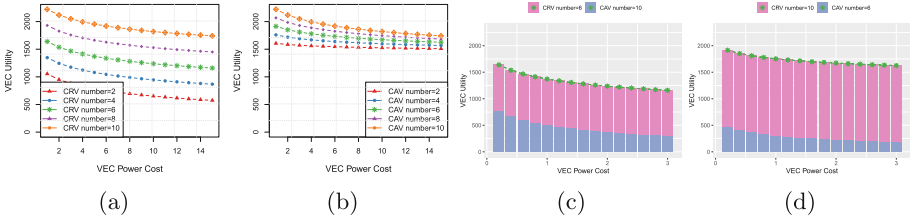


Fig. 3. Effect of VEC power cost on profit of VEC server

The CRV computation cost is also explored if it determines the performance of the proposed model. It can be observed in Fig.4(a) that local computation cost of CRV has no obvious effect on pricing, determined VEC server. With the increase of local computation cost, the steady pricing motivates the CRV to improve the transmit power for reducing local computation consumption in Fig.4(a). We see from Fig.4(b) that CRV utility depends on its computation cost, as well as task transmission time. Given a computation task, shorter task transmission time incurs higher local computation consumption. So, CRV utility tends to dwindle with the increasing local computation cost.

We finally evaluate CAV computation cost on the model performance. Figure4(c) shows the transmit power, allocated by VEC server, and power pricing, determined by CAV. The large CAV computation cost incurs CAV’s high pricing. Accordingly, VEC server reduces the transmit power for utility maximization. Therefore, Fig. 4(c) shows an increasing pricing for VEC power, as well as a decreasing VEC transmit power. Under such scenario, CAV utility gradually degrades with the increasing CAV computation cost in Fig. 4(d). While the power pricing is large enough, the VEC power decreases to zero. CAV utility accords with being zero. Given the computation cost, long task transmission time for CAV promotes the improvement of power pricing and transmit power, on which CAV utility increases.

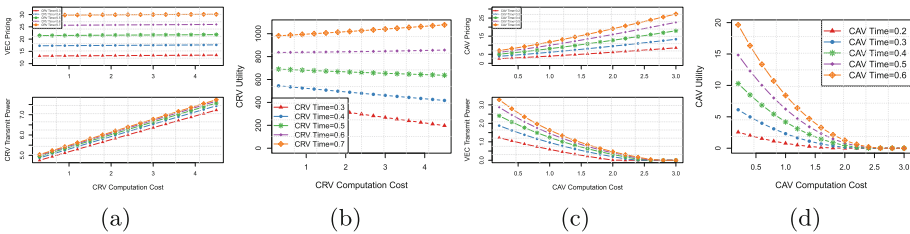


Fig. 4. Effect of CRV and CAV computation cost on utility of VEC server

4 Conclusion

In this paper, we study the incentive mechanism of task offloading based on vehicle edge computing. Vehicles, constrained by limited computation resources, request VEC server for executing computation tasks. To avoid more burdens on VEC server, vehicles with free resources are recruited to assist VEC server's computation work. We are motivated to quantify their gains and costs by defining utility function. Specially, the establishment of utility function considers the modes of task transmission and task computation among them, which is aimed to accurately describe their profits. Moreover, we build resource management and pricing framework by tools of Stackelberg game to ensure that each participant has appropriate incentives to participate in task offloading. Simulation results show that our incentive framework is stable and effective. In the future work, we will concentrate on computation resource management in extensive edge computing networks.

References

1. Raza, S., Wang, S., Ahmed, M., Anwar, M.R.: A survey on vehicular edge computing: architecture, applications, technical issues, and future directions. *Wirel. Commun. Mob. Comput.* **2019**, 3159762:1–3159762:19 (2019)
2. Hou, X., Li, Y., Chen, M., Wu, D., Jin, D., Chen, S.L.: Vehicular fog computing: a viewpoint of vehicles as the infrastructures. *IEEE Trans. Veh. Technol.* **65**, 3860–3873 (2016)
3. Boussehama, M., Benamar, N., Addaim, A.: A new security mechanism for vehicular cloud computing using fog computing system. In: 2019 International Conference on Wireless Technologies, Embedded and Intelligent Systems (WITS) pp. 1–4 (2019)
4. Sun, F., et al.: Cooperative task scheduling for computation offloading in vehicular cloud. *IEEE Trans. Veh. Technol.* **67**, 11049–11061 (2018)
5. Zhang, L., Zhao, Z., Wu, Q., Zhao, H., Xu, H., Wu, X.: Energy-aware dynamic resource allocation in UAV assisted mobile edge computing over social internet of vehicles. *IEEE Access* **6**, 56700–56715 (2018)
6. Hochstetler, J., Padidela, R., Chen, Q., Yang, Q., Fu, S.: Embedded deep learning for vehicular edge computing. In: 2018 IEEE/ACM Symposium on Edge Computing (SEC), pp. 341–343 (2018)
7. Hao, Y., Chen, M., Hu, L., Hossain, M.S., Ghoneim, A.: Energy efficient task caching and offloading for mobile edge computing. *IEEE Access* **6**, 11365–11373 (2018)
8. Qian, J., Duan, B., Xie, W., Zhao, Y.: Edge computing for brightness and color temperature of smart streetlight. In: 2021 IEEE 2nd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE), pp. 699–703 (2021)
9. Shang, B., Zhao, L., Chen, K.C., Chu, X.: An economic aspect of device-to-device assisted offloading in cellular networks. *IEEE Trans. Wireless Commun.* **17**, 2289–2304 (2018)
10. Shi, L., Zhao, L., Zheng, G., Han, Z., Ye, Y.: Incentive design for cache-enabled d2d underlaid cellular networks using Stackelberg game. *IEEE Trans. Veh. Technol.* **68**, 765–779 (2019)



A Dependency-Aware Task Offloading Strategy in Mobile Edge Computing Based on Improved NSGA-II

Chunyue Zhou, Mingxin Zhang, Qinghe Gao^(✉), and Tao Jing

School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing, China
{chyzhou, 20121825, qhgao, t.jing}@bjtu.edu.cn

Abstract. Rapid development of mobile communications has led to respectable latency-sensitive and computation-intensive mobile applications. There is a huge contradiction between high resource demands of these applications and limited resource of mobile devices. In this regard, mobile edge computing (MEC) is a promising technology, where computation tasks can be offloaded from mobile devices onto network edges with stronger capability. However, the dependency between tasks leads to high complexity for offloading decision. In this paper, we investigate the optimal offloading problem for completing dependency-aware tasks by minimizing the latency and energy cost. An improved non-dominated sorting genetic algorithm-II (INSGA-II) is proposed to solve this multiobjective problem. Simulation results validate the advantage of the proposed algorithm in terms of the performance of low latency and cost.

Keywords: Mobile edge computing (MEC) · Task dependency · NSGA-II

1 Introduction

Numerous data computation of applications exceeds the capabilities of mobile devices [1–5]. In this regard, European Telecommunications Standard Institute (ETSI) proposed mobile edge computing (MEC) in 2014 [6]. Closer-distance computation offloading offloads computational-intensive tasks from mobile devices to edge nodes for execution at lower latency and energy cost. Thus, making offloading strategy is significant.

In order to reduce the burden of resource-limited mobile devices, numerous researches have been done on the MEC in recent years. Lu *et al.* replaced edge server resource allocation on blocks of the task with allocation on demand, improving computation resources utilization [7]. Their arbitrary division of task and simultaneous execution can reduce latency, but it is too ideal. An *et al.* considered the sequential dependency between tasks, and reduced the total energy consumption for both slow fast fading channels [8]. Besides, Al-Habob *et al.* considered both sequential and parallel task offloading, and applied genetic algorithm and conflict graph models to obtain minimal latency-reliability cost [9]. Two task division types above consider only execution order or latency reduction, therefore, combining them would be better. Pan *et al.* took the method of model-free approach based on reinforcement learning [10], Liu *et al.* and Chai *et al.* proposed the strategies of prioritizing the multiple tasks in computation

scheduling and offloading [11, 12], Wang *et al.*, Lee *et al.* and Song *et al.* all adopted the heuristic algorithm to reduce the complexity in optimization [13–15], and Zhao *et al.* proved that no algorithm with constant approximation can solve the NP-hard problem, so they designed a convex programming algorithm [16]. All these works above focused the single latency optimization objective, some works, therefore considered multiobjective optimization, such as both Liu *et al.* and Sun *et al.* aimed at both time and energy cost optimization, but applied a distributed deep reinforcement learning strategy and genetic algorithm-based flexible computation offloading and task scheduling algorithm, respectively [17, 18]. Both works took the same method of weighted addition in multiobjective optimization, but for optimization objectives of different dimensions, each objective weight is hard to determine. Unlike the works above, Zhang *et al.* combined Gaussian hidden Bayesian algorithm with KNN algorithm to simultaneously achieve feasibility, effectiveness, and efficiency in offloading [19]. Lu *et al.* also proposed a clustering-assisted offloading scheme based on mobile prediction, resulting in enhancement in the data processing capability of power-constrained networks [20].

This paper, therefore, takes task dependency into account. Meanwhile, the goal of the work is to identify the offloading strategy with minimized latency and cost, which enhance privacy preservation, under the constraints of service configuration and task dependencies [21, 22]. The main contributions of this paper are summarized as followed:

- Dependencies between subtasks is shown by directed acyclic graph (DAG).
- To strike a balance between latency and cost, an optimization problem is formulated to minimize both latency and cost under dependency constraint.
- To solve the non-convex problem, an improved solution based on Non-dominated Sorting Genetic Algorithm II (NSGA-II) is proposed.

The remainder of this paper is organized as follows. The system model and problem formulation are discussed in Sect. 2. The proposed algorithm are clarified in Sect. 3. The simulation results are shown in Sect. 4. The conclusion is drew in Sect. 5.

2 Network Model and Problem Formulation

In this section, the system model is firstly introduced. Then the latency and cost model of task completion are described. Finally, an optimization problem is formulated.

2.1 System Model

A mesh network with a user equipment (UE) and multiple edge nodes (ENs) is considered in this paper, and they can communicate with each other directly and stably as shown in Fig. 1(a). The UE has a task divided into dependent subtasks. Due to resource-limited UE, all subtasks are offloaded onto ENs, then ENs return the final result to UE.

2.2 Task Graph

A UE task is described by a DAG, $G(\mathcal{V}, \mathcal{E})$, as shown in Fig. 1(b), where $\mathcal{V} = \{i_1, i_2, \dots, i_L\}$ is the set of subtasks, and \mathcal{E} is the set of edges representing subtask

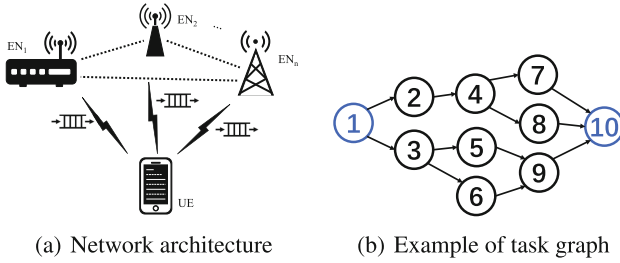


Fig. 1. System model

dependency. The edge $e_{ii^+} = 1$ implies that subtask i is a predecessor of subtask i^+ , and denote transmitted data between them as ω_{i,i^+} . The set of ENs is denoted by $\mathcal{M} = \{m_1, m_2, \dots, m_N\}$. Denote subtask offloading strategy as a matrix $X_{L \times N}$, where element $x_{i,m} = 1$ indicating that subtask i is offloaded onto EN m , otherwise, $x_{i,m} = 0$. Note that a subtask can only be offloaded onto one EN at a time, thus, $\sum_{m \in \mathcal{M}} x_{i,m} = 1$ for $\forall i \in V$.

2.3 Latency Model

In this subsection, the total latency for completing a task is analyzed, including the computation latency, the communication latency and the queuing latency.

- **Computation Latency.** Denote the computation data of each subtask i as ω_i and the computing capacity of EN m as f_m , thus, the computation latency of subtask i executing on EN m is $T_{i,m}^{cmp} = \frac{\omega_i}{f_m}, i \in V, m \in \mathcal{M}$.
- **Communication Latency.** Assume that constant transmission rate between two ENs is r_0 . Then, communication latency between subtask i^- and subtask i from EN m^- to EN m is $T_{i^-,m^-,m}^{cmm} = \begin{cases} 0 & m^- = m \\ \frac{\omega_{i^-,i}}{r_0} & m^- \neq m. \end{cases}$
- **Queuing Latency.** The queuing latency for a subtask i occurs when i) assigned EN is executing other subtasks; ii) predecessor subtasks have not been completed.
- **Total Latency.** The subtask i can start executing on EN m when i) all its predecessor subtasks have been completed; ii) its input data have been transmitted to EN m ; iii) EN m is idle. Thus, actual starting time instant of subtask i at EN m is

$$t_{i,m}^{start} = \max \left(\max_{i \in Pre(i)} \left(t_{i^-,m^-}^{start} + T_{i^-,m^-}^{cmp} + T_{i^-,i,m^-}^{cmm} \right), t_m^{idle} \right), \quad (1)$$

where $Pre(i)$ is the set of immediate predecessors of subtask i , and t_m^{idle} is the time instant that EN m is idle. Then the finishing time instant for the subtask i is $t_i^{finish} = t_i^{start} + T_{i,m}^{cmp}$. Thus, the task completion time can be described as

$$T = \max_{i \in V} t_i^{finish}. \quad (2)$$

2.4 Cost Model

The computation cost of subtask i at EN m can be described as $C_{i,m}^{cmp} = \omega_i \cdot (f_m)^2$. And communication cost between two subtasks from EN m to EN m^+ can be calculated as $C_{ii^+,mm^+}^{cmm} = p_m \cdot T_{ii^+,mm^+}^{cmm}$, where p_m is the transmission power of EN m . Therefore, the accumulative energy consumption for task execution $Cost$ is given as

$$Cost = \sum_{i \in \mathcal{V}} C_{i,m}^{cmp} + \sum_{\substack{i, i^+ \in \mathcal{V} \\ m, m^+ \in \mathcal{M}}} C_{ii^+,mm^+}^{cmm}. \quad (3)$$

2.5 Problem Formulation

Our aim is to minimize the latency by as low energy cost as possible under the constraint of subtask dependency. Thus, we formulate the following optimization problem:

$$\min_{X_{L \times N}} \{T, Cost\} \quad (4)$$

$$s.t. \quad x_{i,m} \in \{0, 1\}, \quad \forall i \in \mathcal{V}, m \in \mathcal{M} \quad (4a)$$

$$\sum_{m \in \mathcal{M}_i} x_{i,m} = 1, \quad \forall i \in \mathcal{V} \quad (4b)$$

$$t_i^{start} + \sum_{m \in \mathcal{M}} x_{i,m} T_{ct} \leq t_{i^+}^{start}, \quad \forall (i, i^+) \in \mathcal{E} \quad (4c)$$

$$t_i^{start} \geq 0, \quad \forall i \in \mathcal{V} \quad (4d)$$

where $T_{ct} = T_{i,m}^{cmp} + \sum_{m^+ \in \mathcal{M}} x_{i^+,m^+} T_{ii^+,mm^+}^{cmm}$. The minimization objectives are latency and energy cost. Constraint (4a) is the indicator between subtasks and ENs; (4b) shows that each subtask can only be offloaded to one EN; (4c) represents task dependency.

The formulated problem is a 0–1 integer programming problem with non-convex constraints. In the following section, we propose an improved non dominated sorting genetic algorithm-II (INSGA-II) to solve this multiobjective problem.

3 Algorithms Design

In this section, we give the details of the INSGA-II, which is different from the conventional NSGA-II in terms of three aspects.

3.1 The Design of INSGA-II

Firstly, we give definitions of three basic elements in a genetic algorithm (GA).

Definition 1. A gene shows the index of the selected EN for a subtask, and multiple genes consist of a chromosome, i.e. a solution to the problem (4). Multiple chromosomes constitute a population with size of N_p . An example is shown in Fig. 2(a).

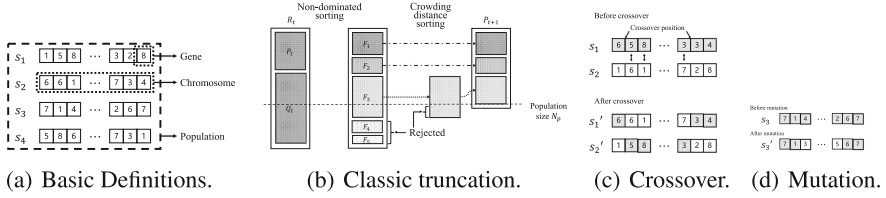


Fig. 2. The key processes of classic NSGA-II.

Then, we introduce each step of the INSGA-II and highlight the solutions for combating the challenges encountered by the traditional NSGA-II algorithm.

- **Step1: Chaotic population initialization.** Initial population in classic NSGA-II is generated randomly, which usually leads to poor search direction and falls into local optima. Lu *et al.* proved that chaotic maps help evolutionary algorithms perform better [23]. In this regard, we employ a chaotic method to generate more evenly distributed initial population as follows: i) Generate a population with size of $2N_p$. Specifically, genes are generated by Logistic map of $x_{k+1} = \mu x_k(1 - x_k)$, $\mu \in [3.57, 4]$, $k = 1, \dots, 2N_p - 1$, where x_1 is a random value from $[0, 1]$; ii) Map fraction of genes into integer EN index. Equidistantly divide range $[0, 1]$ and correspond to available ENs of each subtask, then map the gene fraction into EN index.
- **Step2: On-demand fast non-dominated ranking.** Fast non-dominated ranking in classic NSGA-II as shown in Fig. 2(b) aims at forming intermediate population. The objective values determine the dominance between the chromosomes, which divides chromosomes into various sets, i.e. $F = F_1 \cup F_2 \cup \dots$. Then put N_p chromosomes with smaller ranking into intermediate population. To speed up ranking process, we propose on-demand ranking shown in Algorithm 1, where the ranking process stops once the number of ranked chromosomes is enough.
- **Step3: Crowding distance sorting.** Crowding distance is introduced for a more uniform solution population in the target space, so larger distance is better. The crowding distance of each chromosome is related to adjacent chromosomes with the same non-dominated ranking, and represented below, where f_l and f_c represent latency and energy cost calculation function, respectively.

$$i_d = \begin{cases} \infty & i = 1, N_p \\ \frac{f_l(i+1) - f_l(i-1)}{f_l^{max} - f_l^{min}} + \frac{f_c(i+1) - f_c(i-1)}{f_c^{max} - f_c^{min}} & others. \end{cases} \quad (5)$$
- **Step4: Tournament selection.** Two chromosomes are selected randomly from population, then the chromosome with better non-dominated ranking and crowding distance is put into the parent population with the size of $\frac{1}{2}N_p$.
- **Step5: Crossover with adaptive rate.** SBX performed in NSGA-II may lead to wrong EN indexes. Therefore, this paper applies simple two-point crossover with probability $R_{c,t} = R_{c,max} - (R_{c,max} - R_{c,min})\left(\frac{t}{Gen}\right)^2$, which expands the search solution range upfront and avoids late non-convergence [24].
- **Step6: Mutation with adaptive rate.** Polynomial mutation performed in NSGA-II does not apply. Therefore, multi-point mutation with probability $R_{m,t} = R_{m,min} + (R_{m,max} - R_{m,min})\left(\frac{t}{Gen}\right)^2$ is apply to avoid falling into local optimal solutions.

Algorithm 1. On-demand fast non-dominated ranking**Input:**Population P , population size N_p **Output:**Population with non-dominated rankings F

- 1: For each chromosome p in P , calculate the number of chromosomes that dominate chromosome p (n_p) and the set of chromosomes that chromosome p dominates (S_p)
- 2: Put chromosomes with $n_p = 0$ into set F_1
- 3: $F = F_1, i = 1$
- 4: **while** Population size of $F, N_f < N_p$ **do**
- 5: **for** Chromosome p in set F_i **do**
- 6: **for** Chromosome q in set S_p **do**
- 7: $n_q = n_q - 1$
- 8: **if** $n_q = 0$ **then**
- 9: Put chromosome q into set F_{i+1}
- 10: **end if**
- 11: **end for**
- 12: **end for**
- 13: $F = F \cup F_{i+1}, i = i + 1$
- 14: **end while**
- 15: **if** $N_f = N_p$ **then**
- 16: Keep the population F intact
- 17: **else**
- 18: Calculate crowding distance of chromosomes in set F_i , arrange them in descending order and exclude the last $N_f - N_p$ chromosomes from F
- 19: **end if**

- **Step 7: Elitism selection.** Combine the parent and offspring populations to form a new population, where chromosomes have smaller non-dominated ranking and larger crowding distance.

Repeat the step 2 to step 7 until the end of the iteration.

INSGA-II is extended and enhanced from the conventional NSGA-II by using chaotic population initialization, on-demand non-dominated rankings, and adaptive genetic operators. The general framework of INSGA-II is shown in Algorithm 2.

3.2 Task Offloading

According to the assignment result given by NSGA-II, the earliest start time $EST(i, m)$ and the earliest finish time $EFT(i, m)$ of each subtask i are recorded in a *List*, subtasks in *List* are offloaded in ascending order of their $EST(i, m)$ till *List* is empty. Also, the start time t_i^{start} and finish time t_i^{finish} of each subtask i are updated after offloading.

3.3 Complexity Analysis

Assume that the number of optimization objectives is N_{obj} . Both objective function and crowding distance computation have $\mathcal{O}(N_{obj} \cdot N_p)$ computational complexity.

Algorithm 2. INSGA-II**Input:**Population size N_p , maximum number of iterations of population Gen **Output:**Pareto solutions of offloading strategy P^*

- 1: Initialize population P_0 with size of N_p according to chaotic initialization
- 2: Compute the non-dominated rankings and crowding distance of each solution in P_0
- 3: **for** $t = 1 : Gen$ **do**
- 4: Generate parent population P_t with size of $\frac{1}{2}N_p$ through tournament selection
- 5: Obtain offspring population Q_t with size of N_p by crossing and mutating P_t
- 6: Combine the P_t with Q_t , and obtain population $R_t = P_t \cup Q_t$
- 7: Sort the population R_t according to non-dominated rankings and then crowding distance, then take the first N_p solutions as the new population P_{t+1} through elitism selection
- 8: **end for**
- 9: Find the chromosomes in P_{Gen} with 1st non-dominated ranking as P^* .

Moreover, the computational complexity of Pareto non-dominated sorting is $\mathcal{O}(N_{obj} \cdot N_p^2)$. Thus, the overall complexity of the proposed INSGA-II is $\mathcal{O}(N_{obj} \cdot N_p^2)$.

4 Simulation

4.1 Simulation Setups

In this section, four ENs and a UE task including ten dependent subtasks are considered, and related simulation setups are shown in Table 1. Note that the channel resource and computing rate of each ENs are known in advance.

Table 1. Simulation setups

Parameter	Value
Computation data size of subtask ω_i	[100, 500] KB
Unit computation data amount	[500, 1500] cycles/bit
Computing rate at EN f_m	$[2, 6] \times 10^8$ cycles/s
Transmission power of EN p_m	[0.1, 0.5] W
Transmission rate r_0	866 Mb/s
Ratio of communication to computation data α	0.1
Percentage of ENs configured with required services β	75%
Iteration number	20
Population size	200
Crossover probability	[0.6, 0.8]
Mutation probability	[0.1, 0.3]

To verify the performance of the proposed algorithm, simulations based on randomly generated subtasks in terms of DAG structure is conducted. In addition to conventional NSGA-II, three benchmark algorithms are also introduced.

- **Exhaustion:** Arrange all feasible offloading strategies.
- **Greedy Algorithm:** Select subtasks according to topology sorting of DAG and schedule each subtask to the available EN with the maximum computing capacity.
- **Random Algorithm:** Select subtasks from top to bottom along with the acyclic graph structure and schedule each subtask to a random and available EN.

4.2 Optimization Results

In this section, comparison between multiple algorithm is shown. From the Fig. 3, take the case of only three subtasks as an example, it is clearly seen that INSGA-II has a more equally distributed initial population than that of NSGA-II. Furthermore, the data in Table 2 are obtained when the number of subtasks is 14, the number of ENs is 8 and the population size is 100, which shows that INSGA-II executes more quickly than NSGA-II, especially when the number of iterations increases.

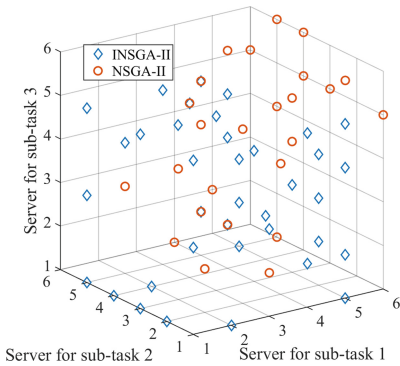


Fig. 3. Comparison of population initialization.

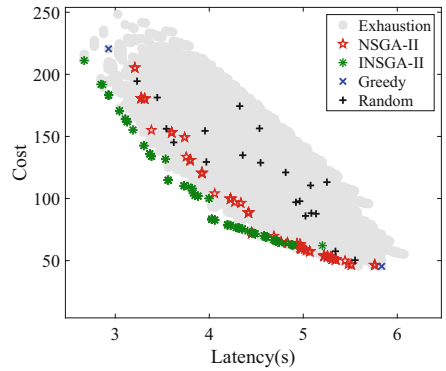


Fig. 4. Comparison of the completion time and accumulative cost.

Table 2. Comparison of algorithm execution time

Number of iteration	10	30	50	70	90
Reduced time (%)	11.16	11.76	16.24	22.43	28.94

Besides, proposed INSGA-II is compared with aforementioned benchmark algorithms and NSGA-II. A point in Fig. 4 represents an offloading solution and shows its latency and cost, figures in terms of latency and cost minimization is shown in Table 3.

Table 3. Numerical optimization results obtained by different approaches

Optimization objective	Latency		Cost	
Algorithm	Objective value			
	Latency(s)	Cost	Latency(s)	Cost
Greedy algorithm	2.929	220.537	5.834	45.396
Random algorithm	3.232	194.486	5.551	50.392
NSGA-II	3.208	205.280	5.765	46.398
INSGA-II-D	2.670	211.119	5.204	62.092

From Fig. 4, it is clearly seen that INSGA-II almost fits the edge of the exhaustion method, which is better than NSGA-II. The greedy algorithm can only obtain the offloading strategy with the lowest accumulative cost, and the random algorithm behaves erratically. The reason why INSGA-II cannot fully fit the edge of the exhaustion method is initial population generation that discards part of cost optimization searching direction. In addition, though the exhaustion method can obtain the best solutions to the formulated problem, its execution time far exceeds the proposed INSGA-II.

5 Conclusion

In this paper, the problem of offloading task divided in the form of DAG is formulated as a multiobjective optimization problem, aiming at reducing latency and cost simultaneously. The proposed INSGA-II effectively reduce the difficulty to obtain a set of optimal offloading strategies. The simulation results demonstrate that INSGA-II outperforms the other algorithms with reduced latency and cost. Future improvement are as follows: 1) consider the INSGA-II of resource competition among multi tasks; 2) use actual data to verify the performance of INSGA-II.

Acknowledgement. This work was supported in part by the Fundamental Research Funds for the Central Universities under Grant 2022JBGP000 and in part by the National Natural Science Foundation of China under Grant 61931001.

References

1. Abbas, N., Zhang, Y., Taherkordi, A., Skeie, T.: Mobile edge computing: a survey. *IEEE Internet Things J.* **5**(1), 450–465 (2018)
2. Cai, Z., Zheng, X., Wang, J., He, Z.: Private data trading towards range counting queries in Internet of Things. *IEEE Trans. Mobile Comput.* 1–17 (2022). Early access
3. Cai, Z., Shi, T.: Distributed query processing in the edge-assisted IoT data monitoring system. *IEEE Internet Things J.* **8**(16), 12679–12693 (2021)
4. Hu, C., Cheng, X., Tian, Z., Yu, J., Lv, W.: Achieving privacy preservation and billing via delayed information release. *IEEE/ACM Trans. Networking* **29**(3), 1376–1390 (2021)
5. Gao, Q., Wang, Y., Cheng, X., Yu, J., Chen, X., Jing, T.: Identification of vulnerable lines in smart grid systems based on affinity propagation clustering. *IEEE Internet Things J.* **6**(3), 5163–5171 (2019)

6. ETSI: Mobile-edge computing introductory technical white paper. White paper (2014)
7. Lu, Y., Zhao, Z., Gao, Q.: A distributed offloading scheme with flexible MEC resource scheduling. In: 2021 IEEE SmartWorld/SCALCOM/UIC/ATC/IOP/SCI, pp. 320–327 (2021)
8. An, X., Fan, R., Hu, H., Zhang, N., Atapattu, S., Tsiftsis, T.A.: Joint task offloading and resource allocation for IoT edge computing with sequential task dependency. *IEEE Internet Things J.* 1–17 (2022). Early access
9. Al-Habob, A.A., Dobre, O.A., Armada, A.G., Muhaidat, S.: Task scheduling for mobile edge computing using genetic algorithm and conflict graphs. *IEEE Trans. Veh. Technol.* **69**(8), 8805–8819 (2020)
10. Pan, S., Zhang, Z., Zhang, T.: Dependency-aware computation offloading in mobile edge computing: a reinforcement learning approach. *IEEE Access* **7**, 134742–134753 (2019)
11. Liu, Y., Wang, S., Zhao, Q., Du, Shiyu, T.: Dependency-aware task scheduling in vehicular edge computing. *IEEE Internet Things J.* **7**(6), 4961–4971 (2020)
12. Chai, R., Li, M., Yang, T., Chen, Q.: Dynamic priority-based computation scheduling and offloading for interdependent tasks: leveraging parallel transmission and execution. *IEEE Trans. Veh. Technol.* **70**(10), 10970–10985 (2021)
13. Wang, M., Ma, T., Wu, T., Chang, C., Yang, F., Wang, H.: Dependency-aware dynamic task scheduling in mobile-edge computing. In: 2020 16th International Conference on Mobility, Sensing and Networking (MSN), pp. 785–790 (2020)
14. Lee, J., Ko, H., Kim, J., Pack, S.: Data: dependency-aware task allocation scheme in distributed edge clouds. *IEEE Trans. Industr. Inf.* **16**(12), 7782–7790 (2020)
15. Song, H., Gu, B., Son, K., Choi, W.: Joint optimization of edge computing server deployment and user offloading associations in wireless edge network via a genetic algorithm. *IEEE Trans. Netw. Sci. Eng.* **9**(4), 2535–2548 (2022)
16. Zhao, G., Xu, H., Zhao, Y., Qiao, C., Huang, L.: Offloading tasks with dependency and service caching in mobile edge computing. *IEEE Trans. Parallel Distrib. Syst.* **32**(11), 2777–2792 (2021)
17. Liu, H., Zhao, H., Geng, L., Wang, Y., Feng, W.: A distributed dependency-aware offloading scheme for vehicular edge computing based on policy gradient. In: 2021 8th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2021 7th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom), pp. 176–181 (2021)
18. Sun, Y., et al.: Dependency-aware flexible computation offloading and task scheduling for multi-access edge computing networks. In: 2021 24th International Symposium on Wireless Personal Multimedia Communications (WPMC), pp. 1–6 (2021)
19. Zhang, P., Zhang, Y., Dong, H., Jin, H.: Mobility and dependence-aware QoS monitoring in mobile edge computing. *IEEE Trans. Cloud Comput.* **9**(3), 1143–1157 (2021)
20. Lu, Y., Chen, Z., Gao, Q., Jing, T., Qian, J.: A mobility-aware and sociality-associate computation offloading strategy for IoT. *Wirel. Commun. Mob. Comput.* **2021**, 9919541:1–9919541:12 (2021)
21. Cai, Z., Zheng, X.: A private and efficient mechanism for data uploading in smart cyber-physical systems. *IEEE Trans. Netw. Sci. Eng.* **7**(2), 766–775 (2020)
22. Cai, Z., He, Z.: Trading private range counting over big IoT data. In: 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), pp. 144–153 (2019)
23. Lu, H., Wang, X., Fei, Z., Qiu, M.: The effects of using chaotic map on improving the performance of multiobjective evolutionary algorithms. *Math. Probl. Eng.* **2014**, 924652 (2014)
24. Deng, Z., Liu, X.: Study on strategy of increasing cross rate in differential evolution algorithm. *Comput. Eng. Appl.* **44**(27), 33–36 (2008)



Federated Reinforcement Learning Based on Multi-head Attention Mechanism for Vehicle Edge Caching

XinRan Li¹, ZhenChun Wei^{1,2}(✉), ZengWei lyu^{1,2}, XiaoHui Yuan³, Juan Xu^{1,2},
and ZeYu Zhang⁴

¹ School of Computer Science and Information Engineering,
Hefei University of Technology, Hefei 230009, China
745161712@qq.com

² Engineering Research Center of Safety Critical Industrial Measurement
and Control Technology, Ministry of Education, Hefei 230009, China

³ Department of Computer Science and Engineering, University of North Texas,
Denton, TX 76203, USA

⁴ Faculty of Information Technology, Macau University of Science and Technology,
Macau 999078, China

Abstract. Vehicles request road condition information, traffic information, and various audio-visual entertainment frequently. Repeat Download will burden the core network and seriously affect the user experience. Edge caching is a promising technology that can effectively alleviate the pressure of repeatedly downloading content from the cloud. There are many existing edge cache scheduling methods, but they all have limitations. his paper proposes an edge cache scheduling method based on the multi-head attention mechanism federal reinforcement learning (FRLMA). Firstly, the problem is modeled as a Markov decision model. The local models are trained through a deep reinforcement learning method. Finally, the federated reinforcement learning framework of edge Cooperative Cache is established. In particular, the multi-head attention mechanism is introduced to weigh the contribution of the local model to the global model from multiple angles. Simulation results show that the FRLMA method has better convergence and is superior to the most current popular methods in terms of hit rate and average delay.

Keywords: Edge caching · Deep reinforcement learning · Federated learning · Multi-head attention

1 Introduction

With the rapid development of the technology of the Internet of vehicles, a large number of innovative vehicle applications have emerged [1], which have a great demand for computing, communication, and storage resources. edge

Supported by the Natural Science Foundation of Anhui Province (2108085MF202).

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022

L. Wang et al. (Eds.): WASA 2022, LNCS 13473, pp. 648–656, 2022.

https://doi.org/10.1007/978-3-031-19211-1_54

caching technology has emerged where popular content is cached at the edge of the network, and users can access and download the cached content directly from the edge [2], which can reduce backhaul data traffic congestion and shorten transmission latency effectively.

Currently, the edge cache scheduling problem has been widely studied by scholars at home and abroad and can be divided into two main categories: traditional and learning-based methods. Most of the existing traditional cache replacement scheduling algorithms are rule-based [3–6], such as first-in-first-out (FIFO), least recently used (LRU), and least recently used. These traditional methods, however, have high complexity, low computational accuracy, and difficulty in adapting to dynamic network environments. Some scholars then used learning methods [7–9] to solve the cache scheduling problem. However, these reinforcement learning methods face the problem of data islands. The federal reinforcement learning method [10] can obtain better training effect while lacking training data. At present, most of the parameters of Federated learning are updated by the method of average aggregation, which is obviously unreasonable. Therefore, this paper mainly studies the edge cache scheduling method based on multi head attention mechanism Federation reinforcement learning. The main contributions of this paper are as follows:

- The content cache scheduling problem in the vehicle Edge network is modeled as a Markov decision process, and the DQN algorithm in reinforcement learning is used to solve the process. The dynamic scheduling of content cache is realized through continuous interaction with the edge network.
- In this paper, a federated parameter aggregation method based on multi-head attention mechanism is proposed and introduced into DQN algorithm to train the local reinforcement learning model in a distributed way. By introducing multi head attention mechanism, the contribution of the local model to the global model is concerned from multiple dimensions, and the problem of parameter aggregation between heterogeneous networks is solved.

2 System Model

2.1 Network Modeling

Network Structure: As shown in Fig. 1, the network architecture is modeled into a three-tier architecture, that are cloud server, base station (BS) and user (UES). Suppose that there are B base stations at the edge, it can be expressed as: $\mathcal{B} = \{b_1, b_2, \dots, b_i, \dots, b_B\}$ each base station has a certain storage capacity, it can be expressed as: $C_c = \{c_1, c_2, \dots, c_i, \dots, c_C\}$ and each base station has a certain number of users in the coverage area, it can be expressed as: $U_i^b = \{u_1, u_2, \dots, u_n\}$. Suppose that there are F kinds of content in the content repository, which can be expressed as $\mathcal{F} = \{f_1, f_2, \dots, f_i, \dots, f_F\}$, in order to make the model more in line with the actual situation, the size of each content is modeled as inconsistent, which is expressed as: $S_f = \{S_{f_1}, S_{f_2}, \dots, S_{f_i}, \dots, S_{f_F}\}$, We order $B * F$ represents the matrix of base station cache status, $\mathcal{H} = (x_{b,f})^{B * F}$, $x_{b_i, f_i} = 1$ indicates the base station b_i has contents f_i , $x_{b_i, f_i} = 0$ indicates base station b_i not have contents f_i .

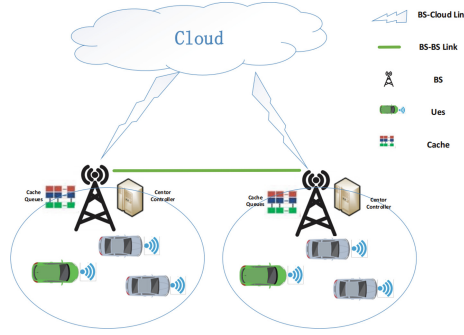


Fig. 1. Edge cache network architecture in a driverless scenario.

Content Popularity: Generally, the probability distribution of content request obeys Zipf distribution, for content f_i , its content popularity can be calculated as follows:

$$\Omega_{f_i} = \frac{r_i^\alpha}{\sum_{i=1}^F r_i^\alpha} \tag{1}$$

where r_i is the content f_i ranking of requested frequency in the content library, α is a constant.

Content Transmission Delay: In this paper, orthogonal frequency division multiple access (OFDMA) is used for communication between the user and the edge server. According to Shannon formula [11], the communication rate between the user and the base station can be expressed as:

$$V_{u,b_i} = w_i \log_2 \left(1 + \frac{P_i g}{\sigma^2} \right) \tag{2}$$

where W_i , P_i , g , and σ^2 represent channel bandwidth, channel transmission power, channel gain, and interference noise power respectively.

Case 1: $T_{f_i,u,b_j} = \frac{S_{f_i}}{v_{u,b_j}}$, $\forall i \in F, \forall j \in B$ indicates that the content requested by the user has hit.

Case 2: $T_{f_i,u,b_j}^O = T_{f_i,u,b_j} + \frac{S_{f_i}}{v_{b,b}}$, $\forall i \in F, \forall j \in B$ Indicates that you need to go to the adjacent base station to obtain content, where, $v_{b,b}$ is the transmission speed between adjacent base stations.

Case 3: $T_{f_i,u,b_j}^O = T_{f_i,u,b_j} + \frac{S_{f_i}}{v_{b_i,c}}$, $\forall i \in F, \forall j \in B$ Indicates that you need to get content from the cloud, where, $v_{b_i,c}$ is the transmission speed between adjacent base stations.

The content transmission delay in the above three cases meets $T_{f_i,u,b_j,c} > T_{f_i,u,b_j}^O > T_{f_i,u,b_j}$.

2.2 Problem Definition

This paper aims to optimize the average delay of content acquisition. The calculation formula of the average delay for n requests is as follows:

$$T_{sum} = \sum_{i \in (N * hit)} T_{f_i, u, b_j} + \sum_{i \in \alpha * (N * (1 - hit))} T_{f_i, u, b_j}^O + \sum_{i \in (1 - \alpha) * (N * (1 - hit))} T_{f_i, u, b_j, c} \quad (3)$$

$$\begin{aligned} P1 \quad & \min \frac{T_{sum}}{N} \\ s.t. \quad & c1 : \alpha \in (0, 1) \\ & c2 : \sum_{t=1}^T R(f_i) \leq C_B \\ & c3 : hit_{ratio} \in (0, 1) \\ & c4 : V_{u,b} > V_{b,b} > V_{b,c} \end{aligned} \quad (4)$$

where, α indicates the proportion of requested content by the local edge server decision to go to adjacent base stations to all content requests, $0 < \alpha < 1$, C2 indicates that the upper limit of the local content cache should be less than the maximum local cache space, C3 indicates that the hit rate of content requests should be between 0 and 1, and C4 indicates that the rate of local transmission is greater than that between adjacent base stations and also between the local and cloud.

2.3 Markov Modeling

We model the physical problem as a Markovian decision process(MDP).

State: $S_t = \{\mathcal{F}_t, \mathcal{H}_t\}$ where \mathcal{F}_t is the content requested by the user in the current time slot, and \mathcal{H}_t is the cache state of the local base station.

Action: $a_t = \{\omega_t^{f_i}, \mathcal{X}_t^{f_i}, A_t^f\}$, where $\omega_t^{f_i}$ is a binary indicator indicating and $x_t^{f_i}$ indicates whether to cache the content f_i in the local cache and A_t^f indicates which local content is replaced.

Reward: The optimization goal of this paper is to minimize the average content acquisition delay of the whole system. Therefore, this paper takes the reciprocal of the transmission delay of the edge device processing N content requests as a reward. Then the immediate reward of the edge device under the time step is:

$$R_t = \frac{N}{\sum_{b_i \in B} T_{b_i}^{sum}} \quad (5)$$

3 Algorithm Design

In order to solve the problems of data shortage and data security, an edge cache scheduling method based on Federated reinforcement learning(FRLMA) is proposed, including local dqn training and cloud parameter aggregation, The whole process of the FRLMA algorithm is shown in Algorithm 1.

3.1 Local DQN Training

Since the action space of the DQN model in this paper is very large, the $Q(s, a)$ of each action cannot be obtained directly. Therefore, assuming $Q(s, a, \theta) \approx Q(s, a)$ and the iterative formula of Q function can be expressed as:

$$Q(s_t, a_t, \theta_t) = (1 - \alpha)Q(s_t, a_t, \theta_t) + \alpha \left(C(s_t, a_t) + \gamma \max_{a_{t+1} \in A_t} Q(s_{t+1}, a_{t+1}, \theta_t^-) \right) \quad (6)$$

where $\alpha \in (0, 1)$ represents the learning rate, when $\alpha = 1$, the Q value is completely updated to the newly calculated Q value. Otherwise, it will be modified and fine-tuned on the basis of retaining part of the historical Q value. θ is the parameter for evaluating the network and θ^- is the parameter of the target network. The loss function of DQN model can be expressed as:

$$Loss(\theta) = E \left[C(s_t, a_t) + \gamma \left(\max_{a_{t+1} \in A_t} Q(s_{t+1}, a_{t+1}, \theta_t^-) - Q(s_t, a_t, \theta_t) \right)^2 \right] \quad (7)$$

parameter θ The update expression of can be expressed as:

$$\theta_{t+1} = \theta_t - \alpha \nabla(Loss(\theta)) \quad (8)$$

3.2 Parameters Aggregation

This section proposes a parameter aggregation method based on a multi-headed attention [12] mechanism weighted by the following procedure.

Local Environmental Status of Participants: Average reward: the average reward is defined as the average value of the reward $C(s, a)$ of the local model in the local M round training, which can be expressed as C_{ave} ; Local cache capacity: the larger the local cache capacity, the more content can be cached locally, and the cache hit rate will also increase, The local cache capacity can be defined as D_{max} ; The parameters of Zipf distribution: The parameters of Zipf distribution determines the density of content requests, the larger the parameter, the greater the cache hit rate. Zipf parameters can be defined as zip_{α} ; Data set size: the local training data of each local model is different, and the size of the data set can be expressed as S_c .

Local Environmental Status of Participants: Experience pool size: for those devices with more memory resources, they can store more training data into replay memory and thus learn more about previous experience, the size of the experience pool can be expressed as R_c ; Batch size: As each local model has different computational power, some local models have the more computational power and can train more data in one local training process, denoted as B_c . The above evaluation metric can be expressed as $K_b = [C_{ave}^b, D_{max}^b, zip_{\alpha}^b, S_c^b, R_c^b, B_c^b]$, in order to obtain a better local model, more rewards, shorter latency and higher cache hit ratio, we design the query as $Q = [\max(C_{ave}^b), \max(D_{max}^b), \max(zip_{\alpha}^b), \max(S_c^b), \max(R_c^b), \max(B_c^b)]$, the specific aggregation process is as follows:

- 1) Split operation: The ECS determines the number of heads to be split, and splitting the local inputs K_b , Q by a linear transformation:

$$K_b^i = \text{linear}(K), \quad Q_b^i = \text{linear}(Q) \quad (9)$$

- 2) Attention operation: The attention mechanism is manipulated for each head to obtain the weight of each head:

$$\omega_{head_i} = \text{softmax} \left(\frac{Q_b^i K_b^i T}{\sqrt{\frac{d_k}{h}}} \right) \quad (10)$$

- 3) Concat heads: The weights of each head are combined by a weight matrix to obtain the final weights for each local model b :

$$\omega_b = \text{concat}(\omega_{head_1}, \omega_{head_2} \dots \omega_{head_h}) \quad (11)$$

ω_b can be seen as a measure of the magnitude of the contribution of the local model to the global model, so the update parameter Θ of the global model can be calculated by:

$$\Theta = \sum_{b \in B} \omega_b \theta_b \quad (12)$$

Algorithm 1. Framework of FRLMA.

Input: Network Status: $S_t = \{\{f_{i,t}, S_{f_{i,t}}, \Omega_{i,t}\}, \mathcal{H}_t\}$;

Output: Q value for each action $Q(s_t, a_t, \theta_t)$;

- 1: Initialisation: Local network parameters; Number of iterations n ; Updates frequency M .
- 2: **for** $each \in [1, B]$ **do**
- 3: **for** $episode \in [1, n]$ **do**
- 4: Local BS gets content requests f from UES;
- 5: **if** $x_{b,f} = 1$ **then**
- 6: break;
- 7: **else**
- 8: **if** $C_b - \sum_{i=1}^F x_{b,f_i} \times S_{f_i} \geq S_f$ **then**
- 9: Cache content f to local BS
- 10: **else**
- 11: Observe
 the current state of the environment $S_t = \{\{f_{i,t}, S_{f_{i,t}}, \Omega_{i,t}\}, \mathcal{H}_t\}$
- 12: Select action $a_t = \arg \max Q(s_t, a_t, \theta_t)$
- 13: Execute action a_t
- 14: Obtain immediate reward $r_t = C(s_t, a_t)$
- 15: Observe the new state s_{t+1}
- 16: Store (s_t, a_t, r_t, s_{t+1}) into the experience pool

```

17:         if  $n\%M = 0$  then
18:             Send local model parameter to the cloud for aggregation
19:             Obtain the parameters  $\Theta$  after cloud aggregation
20:             Update local network parameter  $\theta = \Theta$ 
21:         else
22:             Randomly sample a mini-batch  $(s_t, a_t, r_t, s_{t+1})$  from the experi-
23:             ence pool
24:             Update the parameters of the evaluation network  $\theta_{t+1} = \theta_t -$ 
25:              $\alpha \nabla(Loss(\theta))$ 
26:             Update target network parameter periodically  $\theta^- = \theta$ 
27:         end if
28:     end for
29: end for

```

4 Simulation

In this section, we first describe the setting of the experimental parameters, then verify the convergence of the algorithm, and finally compare the performance with other algorithms.

4.1 Experimental Parameter Settings

We set the number of base stations participating in the federation to 3, the number of contents of the whole system to 2000, and then set the size of each content to 5–15 M. According to the Mzipf distribution [13], this paper uses three values of 0.7, 0.9 and 1.1 to set the popularity of the contents, the storage space size of local base stations is taken as 200–1000 M, the transmission rate between local base stations is 50 M/s, the transmission rate between local base stations to the cloud is 10 M/s, the bandwidth of RSU is 100 MHz, the transmission power is 35 dBm, the Gaussian noise power is -95 dBm, and the information gain of this paper is set to 100. In addition, we set the experience pool size of the three local network models participating in the federation to 1000–2000, as well as different batch sizes of 32–64 for each local base station. The deep networks used in the local models were neural networks with two hidden layers, and the number of neurons in the hidden layer was 64. The learning rate of the local networks was 0.01, and the value of e-greedy was 0.9.

4.2 Algorithm Performance Comparison

To verify the performance of the FRLMA under different data, we compare it with the first-in-first-out algorithm (FIFO), least recent unused (LRU), least recent used (LFU), deep reinforcement learning (DQN), federal average based

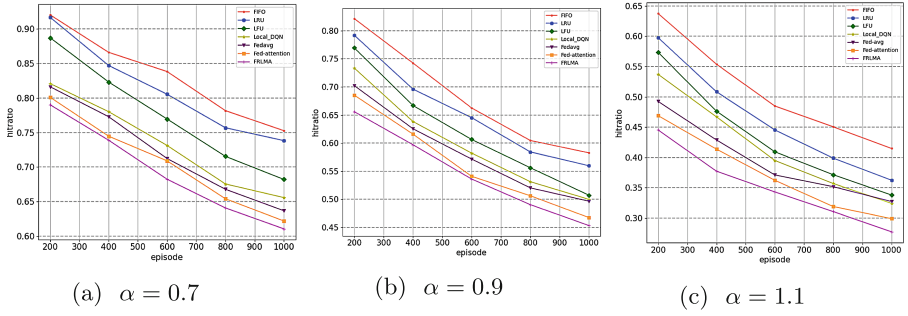


Fig. 2. Comparison of the average time delay between FRLMA and other six algorithms with different Zipf parameters.

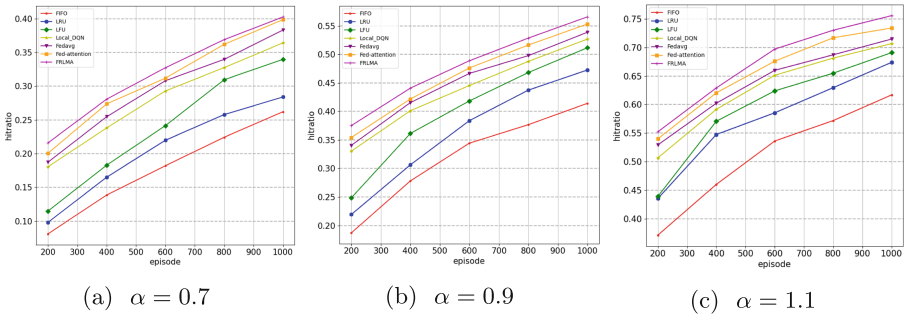


Fig. 3. Comparison of the cache hit rate of FRLMA and six other algorithms with different Zipf parameters.

reinforcement learning (Fedavg), federal reinforcement learning based on the self-attention mechanism (Fed-attention), and we will evaluate the performance of the algorithm of the article in terms of the following two metrics: 1) cache hit rate and 2) average latency.

Figures 2 and 3 show the comparison of the cache hit rate and average latency between FRLMA and the other six algorithms for parameters α of 0.7, 0.9, and 1.1, respectively. From Fig. 2, we can see that the FRLMA always has a higher cache hit rate than the other six algorithms. Figure 2(c) shows that when the parameter α of Zipf distribution is 1.1 and the storage space of the BS is 1000M, the cache hit rate of the FRLMA reaches 75% and the average fetching latency of the task is about 0.3s, which is about 15% better than the traditional LRU algorithm and the average latency is reduced by about 0.15s. Compared to the centralized DQN algorithm, the cache hit rate is improved by about 5% and the average content fetching latency is reduced by 0.1 s. Compared to the other two federal reinforcement learning algorithms, the FRLMA is also improved in terms of both the cache hit rate and the average content fetching latency.

5 Conclusion

Aiming at the problem of multi edge secure Cooperative Cache, this paper establishes a three-level cache architecture between users and edge devices, between edge devices, and between edge devices and ECs, and establishes an optimization problem model with the goal of minimizing the average delay and maximizing the content hit rate. By introducing the federal learning framework, an edge cache scheduling method (fmlma) based on multi head attention mechanism federal reinforcement learning is proposed, which improves the identification ability of the contribution of individuals participating in federal learning to the learning model. Through simulation experiments, the advantages of the proposed algorithm in request hit rate, average delay and convergence speed are verified.

References

1. Ren, K., Wang, Q., Wang, C., Qin, Z., Lin, X.: The security of autonomous driving: threats, defenses, and future directions. *Proc. IEEE* **108**(2), 357–372 (2020)
2. Yang, W., He, S., Fan, X., Xu, C., Sun, X.H.: On cost-driven collaborative data caching: a new model approach. *IEEE Trans. Parallel Distrib. Syst.* **30**, 662–676 (2018)
3. Li, K., Kumar, S., Lloyd, W.A., Huang, Q., Tang, L.: RIPQ: advanced photo caching on flash for facebook. *USENIX Association* (2015)
4. Jia, W.: A survey of web caching schemes for the internet. *ACM SIGCOMM Comput. Commun. Rev.* **29**(5), 36–46 (2000)
5. Rossi, D., Rossini, G., Paristech, T., Paris, F.: Caching performance of content centric networks under multi-path routing (and more) (2011)
6. Zhao, J., Sun, X., Li, Q., Ma, X.: Edge caching and computation management for real-time internet of vehicles: an online and distributed approach. *IEEE Trans. Intell. Transp. Syst.* **22**(4), 2183–2197 (2021)
7. Pang, H., Liu, J., Fan, X., Sun, L.: Toward smart and cooperative edge caching for 5G networks: a deep learning based approach. In: 2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS) (2018)
8. Hasselt, H.V., Guez, A., Silver, D.: Deep reinforcement learning with double q-learning. *Computer Science* (2015)
9. Li, R., Zhao, Y., Wang, C., Wang, X., Taleb, T.: Edge caching replacement optimization for d2d wireless networks via weighted distributed DQN. In: 2020 IEEE Wireless Communications and Networking Conference (WCNC) (2020)
10. Bonawitz, K., et al.: Towards federated learning at scale: system design (2019)
11. Tang, H., Ding, Z.: Mixed mode transmission and resource allocation for d2d communication. *IEEE Trans. Wirel. Commun.* **15**(1), 1–1 (2016)
12. Vaswani, A., et al.: Attention is all you need. *arXiv* (2017)
13. Hefeeda, M., Saleh, O.: Traffic modeling and proportional partial caching for peer-to-peer systems. *IEEE/ACM Trans. Netw.* **16**(6), 1447–1460 (2008)



Research on NER Based on Register Migration and Multi-task Learning

Haoran Ma^{1,2}, Zhaoyun Ding¹(✉), Dongsheng Zhou³, Jinhua Wang²,
and ShuoShuo Niu²

¹ Science and Technology On Information Systems Engineering Laboratory, National University of Defense Technology, Chang Sha, China

zyding@nudt.edu.cn

² The 32Nd Research Institute of China Electronics Technology Group Corporation, ShangHai, China

³ National and Local Joint Engineering Laboratory of Computer Aided Design, School of Software Engineering, Dalian University, Dalian, China

Abstract. The insufficient number of tags is currently the biggest constraint on named entity recognition (NER) technology, with only a small number of Registers (means the domain of language, which will be explained in Part I) currently having a corpus with sufficient tags. The linguistic features of different Registers vary greatly, and thus a corpus with sufficient labels cannot be applied to NER in other Registers. In addition, most of the current NER models are more designed for large samples with sufficient labels, and these models do not work well in small samples with a small number of labels. To address the above problems, this paper proposes a model T_NER based on the idea of migration learning and multi-task learning, which learns the common features of language by using the idea of multi-tasking, and passes the model parameters of neurons with common features of language learned from multiple well-labelled source domains to the neurons in the target domain to achieve migration learning based on parameter sharing. In baseline experiments, T_NER's neurons outperformed the original models such as BiLSTM and BiGRU on a small-sample NER task; in formal experiments, the more the Registers in source domains, the better T_NER's recognition of the target domain. The experiments demonstrate that T_NER can achieve NER for small samples and across Registers.

Keywords: NER · Register · Transfer learning · Multi-task learning

1 Introduction

NER is one of the most fundamental of natural language processing techniques, which belongs to the problem of sequence labeling [1], a number of methods have been proposed for named entity recognition [2–4], and although these methods have improved the accuracy of named entity recognition, these models still rely on large-sample corpus and cannot be analyzed and studied on small-sample corpus. The distinction between large and small samples is based on the number of tags in the text, the number of tags

in small samples are usually in the order of 100,000 and above, while the number of tags in small samples is less than 10,000. The current mainstream models are poor at recognising named entities in small- sample corpus training (see Sect. 4 for details).

Register is a conception belongs to philology category, which means the language domain. [13] The distinction between Registers is usually based on two kinds: firstly, language texts from different domains can be regarded as different Registers, for example, news texts and advertising texts belong to different Registers; secondly, language texts from the same domain from different eras can also be regarded as different Registers, for example, news reports from different eras belong to different Registers. The distribution characteristics of named entities in different Registers will be different.

The current reality is that most Registers' texts have a small number of labels, and only a small number of Registers have a sufficient number of labels; it is not possible to use a corpus with sufficient labels to train a corpus with insufficient labels (because named entities across Registers do not work well).

In order to solve the problems in the field of NER, this paper proposes the model T_NER, which is a new NER model. Firstly, the model is based on Register migration, by learning the source domain corpus, adjusting the model parameters, and applying the adjusted model to the training of the target domain to achieve migration learning based on parameter sharing. The model also utilises the idea of multi-task learning by using different corpus as the source domain and learning them simultaneously, thus extracting the common features of the language and applying them to the recognition of named entities in the small sample target domain.

The paper is organized as follows: Sect. 2 provides a summary of cutting-edge methods in the relevant field, Sect. 3 provides a detailed analysis of the model architecture, describes and analyses the experiments involved in the model, and Sect. 4 summarizes and analyses the model as well as the experiments.

2 Related Work

2.1 NER Techniques

NER is the most fundamental technique in the field of natural language processing. In addition to deep learning, many new approaches and ideas have been applied to the problem of NER: MetaNER uses the technical approach of meta-learning [2], which makes the model learns and analyses the corpus faster and more efficiently; BERT- BiLSTM-GAM-CRF uses the approach of syntactic analysis [3], which focuses on relying on linguistic logic to redefine features; besides, there are new model design ideas using attention mechanisms or multi-tasking [4], all of which are good at learning features of the language.

2.2 Transfer Learning Techniques

Transfer learning is a method of machine learning that refers to a pre-trained model being reused in another task, the effect of one type of learning on another, or the effect of an acquired experience on the completion of other activities.

Migration learning is currently widely used in the following areas: feature-based migration for cross-domain gesture recognition [5]; instance-based migration for radioparticle tracking [6]; and optimization of reinforcement learning and meta-inspired optimization algorithms using migration learning [7].

Currently, migration learning is also widely used in the field of NER: TL-NER, which combines migration learning with deep learning for Chinese NER [8]; BioNER, based on a migration learning paradigm, investigates entity relationships in the medical domain [9]; Trans-NER, a feature-based migration learning, character-level vector generation algorithm for contextual features to migrate source model knowledge to entity recognition models [10].

2.3 Multi-task Learning

Multi-task learning is one of the most popular machine learning methods, which aims to perform simultaneous learning of multiple projects, using parameter sharing to better enable learning for different tasks.

Multi-task learning has a wide range of applications, and MT-DNNs perform multiple specific tasks in the pre-training phase to better learn the features of the language [11]. Zhan Fei et al. used multi-task learning to achieve model building for short textual entity links [12].

3 Model

3.1 Related Definitions

T_NER is a small-sample oriented, migration learning based NER model using the idea of multi-task learning. The model aims to recognise named entities in small samples of inadequately annotated corpus by learning from multiple adequately annotated corpora of different Registers, based on the common features of the learned languages.

Formally, the source domain of T_NER is $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$ and the target domain is (x_t, y_t) ($t \neq 1, 2, \dots, n$). Where x represents the character text and the corresponding y represents the named entity label corresponding to the character. The (x_i, y_i) ($i = 1, 2, \dots, n$) in the source domain represents a sufficiently labeled large sample corpus from different domains; the (x_t, y_t) in the target domain represents a small sample corpus different from each training sample Register in the source domain.

3.2 Model Structure

T_NER is divided into two parts, the first part is the source domain training structure and the second part is the target domain extraction structure, the overall structure of the model is shown in the following figure (Fig. 1):

The corpus in source domain and target domain belong to different Registers. How to extract the common features of the language through many different Registers. This model is designed with the thought of multi-task. T_NER treats the training of different domains as different tasks, and learns from many different domains (including a large

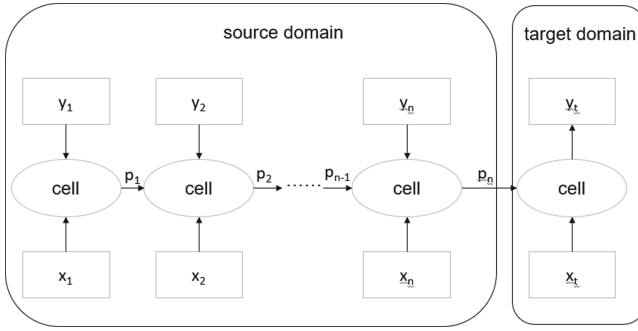


Fig. 1. T_NER model architecture

number of texts and corresponding labels). This learning approach mitigates the overfitting of the model at the Register level due to single Register, the multi-Register mixture training can better highlight the common features of the language.

At the same time, T_NER uses transfer learning based on parameter sharing, which enables the transfer of knowledge between different Registers by sharing training parameters from different Registers.

Analyzed from a structural point of view, at a micro level, the training structure of the source domain is a linear combination of several cells. The cell is a special design structure of T_NER, the mathematical representation is as follows:

$$p_i = \varnothing(x_i, y_i, p_{i-1}) \tag{1}$$

where p_i denotes the output of layer i cell ($i \in [1, n]$) which is the model parameters of layer i cell and denotes the model parameters passed to layer $i + 1$. (x_i, y_i, p_{i-1}) denotes the input to the cell at layer i , x_i denotes the text of group i , y_i denotes the label corresponding to the text of group i , and p_{i-1} denotes the neural parameters passed from the cell at layer $i-1$ to the cell at layer i . \varnothing denotes the function of the source domain cell (in fact this is a description from a mathematical point of view, the essence of this function is the cell model).

For the target domain structure, the logic can be expressed by the following mathematical formula:

$$y_t = \theta(p_n, x_t) \tag{2}$$

x_t denotes the text of the target domain, p_n denotes the model parameters passed from the source domain to the target domain. y_t denotes the label corresponding to x_t , the θ denotes the function of the target domain cell.

3.3 Neuronal Structure

The structure of each cell is shown in the figure below (Fig. 2):

Each cell structure consists of three parts: the pre-training module, the BiLSTM + BiGRU encoder module and the CRF decoder module.

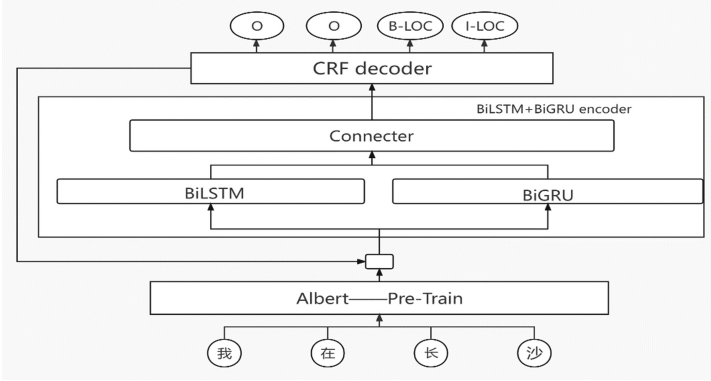


Fig. 2. T_NER neuron architecture

The pre-training module uses the Albert model to transform the text in the source domain from the form of characters to the form of word vectors, which can speed up the subsequent training.

The input of BiLSTM + BiGRU encoder module is the word vector trained by pre-training module, and the output is a probabilistic combination of the corresponding label sequences of the text, i.e. the module encodes the word vector into a probabilistic combination of the corresponding label sequences. It is found that BiLSTM is better at feature learning for long texts and BiGRU is better at feature learning for short texts, so the model combines the two as a learning structure for sequence features.

The BiLSTM + BiGRU encoder is concrete structured as follows (Fig. 3).

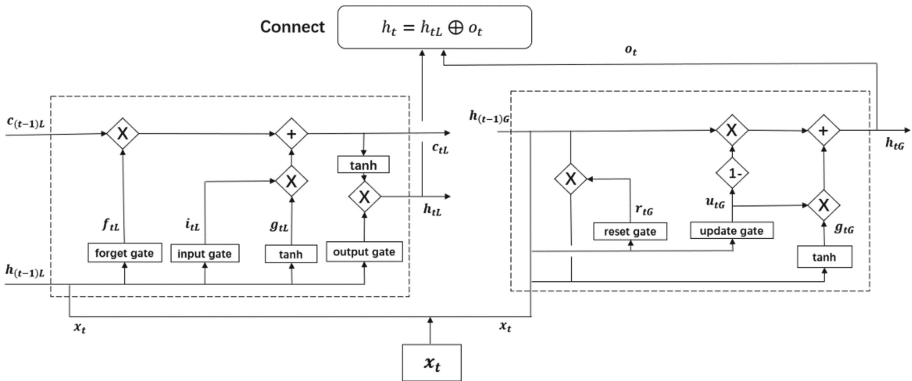


Fig. 3. BiLSTM + BiGRU encoder architecture

The principle of operation can be described by the following equation.:

$$i_{tL} = \sigma(W_i x_t + U_i h_{(t-1)L} + b_i) \tag{3}$$

$$f_{tL} = \sigma(W_f x_t + U_f h_{(t-1)L} + b_f) \tag{4}$$

$$o_{tL} = \sigma(W_o x_t + U_o h_{(t-1)L} + b_o) \quad (5)$$

$$g_{tL} = \tanh(W_g x_t + U_g h_{(t-1)L} + b_g) \quad (6)$$

$$c_{tL} = f_{tL} * c_{(t-1)L} + i_{tL} * g_{tL} \quad (7)$$

$$h_{tL} = o_t * \tanh(c_t) \quad (8)$$

$$r_{tG} = \sigma(W_r x_t + U_r h_{(t-1)G} + b_r) \quad (9)$$

$$u_{tG} = \sigma(W_u x_t + U_u h_{(t-1)G} + b_u) \quad (10)$$

$$g_{tG} = \tanh(W_g x_t + U_g (r_t * h_{(t-1)G})) \quad (11)$$

$$h_{tG} = (1 - u_{tG}) * h_{(t-1)G} + u_{tG} * g_{tG} \quad (12)$$

$$h_t = h_{tL} \oplus h_{tG} \quad (13)$$

In the above equation, σ denotes the sigmoid function, and \oplus denotes the splicing of vectors (i.e. splicing two 1×4 vectors into a 1×8 vector). The subscripts *L denote the parameters and output of the BiLSTM neuron, while the subscripts *G denote the parameters and output of the BiGRU neuron. Where x_t denotes the trained word vector and h_t denotes the probabilistic sequence splicing obtained from the hybrid training, i.e. the linear splicing of the label probabilities obtained by each of BiLSTM and BiGRU.

The CRF decoder module has a CRF as its core structure, the purpose of which is to provide more constraints on the probabilistic combinatorial splicing obtained from the BiLSTM + BiGRU encoder, the essence of which is to decode the probabilistic combinations of label sequences into their corresponding deterministic labels. Experiment.

3.4 Experimental Configuration

Three BIO-tagged corpora were used in experiment: 1998 People's Daily, 2014 People's Daily and sighthan 2006 MSRA (Microsoft Research Asia). There are three categories of entities in the corpus, namely LOC (location category), ORG (organisation category) and PER (person name category).

According to the study, although the three corpora in the experiment all belong to the news category corpus, the three corpora belong to different eras, and therefore the three corpora belong to different Registers.

In this paper, a small sample corpus and a large sample corpus are classified as follows: a small sample is one in which the total number of entities in the text is less than five thousand, and a large sample is one in which the total number of entities in the text is greater than five thousand.

3.5 Experimental Effects Discriminated

This experiment uses $f1_score$'s micro avg, macro avg, and weighted avg as the discriminatory criteria for model effects, calculated as follows.

$$\text{micro R} = (\sum TP) / (\sum TP + \sum FN) \quad (14)$$

$$\text{micro P} = (\sum TP) / (\sum TP + \sum FP) \quad (15)$$

$$\text{micro } f1_score = (2 * \text{micro } P * \text{micro } R) / (\text{micro } P + \text{micro } R) \quad (16)$$

$$\text{precision}_i = (TP_i) / (TP_i + FP_i) \quad (17)$$

$$\text{recall}_i = (TP_i) / (TP_i + FN_i) \quad (18)$$

$$f1_score_i = (2 * \text{precision}_i * \text{recall}_i) / (\text{precision}_i + \text{recall}_i) \quad (19)$$

$$\text{macro } f1_score = (\sum_1^n f1_score_i) / n \quad (20)$$

$$\text{weighted } f1_score = \left[\sum_1^n f1_score_i * (TP_i + FN_i) \right] / \sum_1^n (TP_i + FN_i) \quad (21)$$

TP means that the true category is positive and the predicted category is positive. FP means that the true category is negative and the predicted category is positive. FN means that the true category is positive and the predicted category is negative. TN means that the true category is negative and the predicted category is negative. Micro $f1_score$ \textbackslash macro $f1_score$ \textbackslash micro $f1_score$ refers to the micro avg, macro avg, and weighted avg of $f1_score$, respectively.

3.6 Neuron Utility Experiment

This set of experiments mainly compared the named entity recognition capabilities of BiLSTM + CRF, BiGRU + CRF and T_NER neurons in small samples.

We use the 1998 People's Daily corpus in this part, which was fed into BiLSTM + CRF, BiGRU + CRF and T_NER neurons respectively for training. The results were as follows (Table 1 and Fig. 4):

Table 1. BiLSTM + CRF, BiGRU + CRF, T_NER experimental control results.

Neuron model	Micro avg	Macro avg	Weighted avg
BiLSTM + CRF	0.8037	0.7969	0.8037
BiGRU + CRF	0.8026	0.7966	0.8026
T_NER	0.8247	0.8188	0.8246

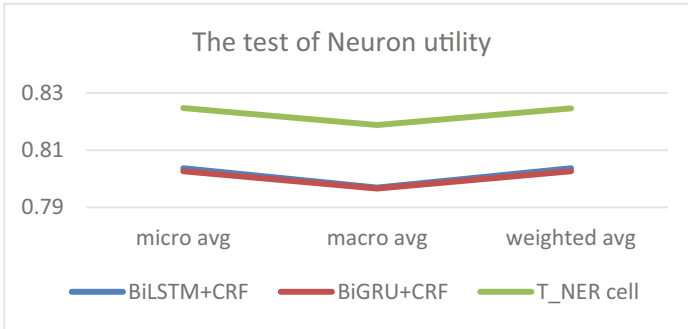


Fig. 4. The comparison of the neuron utility.

From the above table, we can easily find that T_NER neurons outperformed BiLSTM + CRF and BiGRU + CRF in the test, and this experiment demonstrates that T_NER neurons have a greater improvement in NER in small sample domains compared to traditional models.

3.7 Multitasking Experiment

The purpose of this experiment is twofold: firstly, to verify the facilitative effect of the presence of the source domain on the target domain for NER, and secondly, to verify the effect of the amount of corpus in the source domain, i.e. the degree of multitasking, on the effectiveness of NER.

This experiment involves three sets of BIO-tagged corpus, namely 1998 People’s Daily, 2014 People’s Daily and sishan 2006 MSRA, which are in different Registers. The first two serve as the source domain and the latter as the target domain.

This experiment is divided into four groups, the first group is to perform NER on the target domain text only; the second group is to use only the 1998 People’s Daily corpus as the source domain and then perform NER on the target domain; the third group is to use only the 2014 People’s Daily corpus as the source domain and then perform NER on the target domain; the fourth group is to use both the 1998 People’s Daily corpus and 2014 The fourth group uses both the 1998 People’s Daily corpus and the 2014 People’s Daily corpus as source domains, and then identifies named entities for the target domain.

The comparison between the first set of experiments and the fourth set of experiments will verify the facilitative effect of the presence of the source domain on the target domain

for NER. The comparison of the second set of experiments, the third set of experiments and the fourth set of experiments is to verify the effect of the degree of multitasking, on the effectiveness of NER. The results of the four sets of experiments are as follows (Table 2 and Fig. 5).

Table 2. The impact of multitasking on learning outcomes.

Neuron model	Micro avg	Macro avg	Weighted avg
Lab1	0.4039	0.3896	0.4060
Lab2	0.5494	0.5369	0.5527
Lab3	0.6359	0.6226	0.6389
Lab4	0.6518	0.6425	0.6563

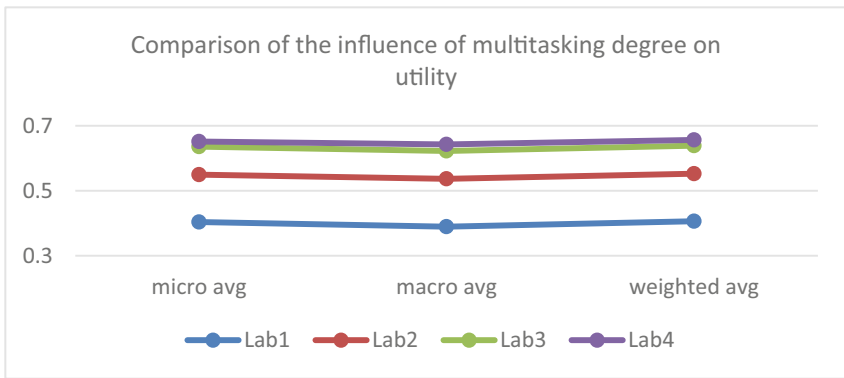


Fig. 5. Comparison of the influence of multitasking degree on utility

From the above experimental results, we can draw two conclusions: 1. The presence of the source domain has a facilitating effect on the NER ability of the target domain. 2. The more corpus in the source domain, the better the NER of the target domain.

4 Summary

Through the different experiments in this article, the excellent results of T_NER in cross-domain NER with small samples are verified on the one hand, and on the other hand, two features in migration learning as well as multi-task learning are verified: the more adequate the source domain in migration learning, the better the experimental results in the target domain; the more tasks in multi-task learning, the better the final learning results.

T_NER still has a number of shortcomings: the BiLSTM and BiGRU encoders improve the experimental results but are difficult to interpret from a mathematical point of view; the training is not fast enough; and the accuracy is not high enough. Despite

these shortcomings, T_NER is still a very good model for NER in terms of robustness and transferability.

References

1. Sun, Z., Wang, H.: Overview on the advance of the research on named entity recognition. New technology of library and information service. Title of a Proceedings Paper. In: Conference 2016. LNCS, vol. 9999, pp. 1–13. Springer, Heidelberg (2016)
2. Li, J., Shang, S., Shao, L.: MetaNER: named entity recognition with meta-learning. In: Proceedings of the 2020 IW3C2 (International World Wide Web Conference Committee) (2020)
3. Li, D., Yan, L., Yang, J., Ma, Z.: Dependency syntax guided BERT-BiLSTM-GAM-CRF for Chinese NER. Expert Syst. Appl. **196**, 116682 (2022)
4. Zhu, P., Cheng, D., Yang, F., Luo, Y., Qian, W., Zhou, A.: ZH-NER: Chinese named entity recognition with adversarial multi-task learning and self-attentions. In: Jensen, C.S., et al. (eds.) DASFAA 2021. LNCS, vol. 12682, pp. 603–611. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-73197-7_40
5. Wu, B.-X., Yang, C.-G., Zhong, J.-P.: Research on transfer learning of vision-based gesture recognition. Int. J. Autom. Comput. **18**(3), 422–431 (2021). <https://doi.org/10.1007/s11633-020-1273-9>
6. Lindner, G., Shi, S., Vuceticb, S., Miskovic, S.: Transfer learning for radioactive particle tracking. Chemical Engineering Science
7. Durgut, R., Aydin, M.E., Rakib, A.: Transfer learning for operator selection: a reinforcement learning approach. Algorithms **15**, 24 (2022)
8. Peng, D., Wang, Y., Liu, C., Chen, Z.: TL-NER: a transfer learning model for Chinese named entity recognition. Inf. Syst. Front. **22**(6), 1291–1304 (2019). <https://doi.org/10.1007/s10796-019-09932-y>
9. Kumar, A., Sharaff, A.: NER based biomedical entities association extraction using transfer learning technique. Mater. Sci. Eng. **1022**, 012055 (2021)
10. Wang, Y., Peng, D., Chen, Z., Liu, C.: Trans-NER: a Chinese Named Entity Recognition Model Supported by Transfer Learning. Journal of Chinese Computer Systems
11. Liu, X., He, P., Chen, W., Gao, J.: Multi-Task Deep Neural Networks for Natural Language Understanding. On the latest GLUE test set (2019)
12. Fei, Z., et al.: A method of short text entity linking based on multi-task learning. Computer Engineering
13. Delu, Z.: Introduction to Domain Theory. Modern Foreign Languages (1987)

Author Index

- Albishari, Mohammed III-563
Al-Hammadi, Ikhlas III-563
Almosharea, Esmail III-563
Amran, Gehad Abdullah III-563
- Bai, Jie III-345
Bai, Yanhong I-216
Ban, Dayan II-343
Bao, Zheng I-113
- Cai, Jun III-20
Cai, Kaiyu I-314
Cai, Lin X. III-541
Cao, Cong I-545
Cao, Hongtang III-238
Cao, Huapeng II-279
Cao, Jin II-406
Cao, Jinhui II-489
Cao, Liyuan I-314
Cao, Sheng III-529
Cao, Yibo I-244, II-356
Cao, Ziwen II-582
Cao, Zouying II-303
Chai, Hua I-185
Che, Yudi II-40
Chen, Guihai II-317
Chen, Jiajun II-514
Chen, Jiewei III-456
Chen, Jing I-153
Chen, Lei II-526
Chen, Liwei I-271
Chen, Ning III-155
Chen, Pengpeng II-105, III-213
Chen, Rongshan II-291
Chen, Sheng III-468
Chen, Tianjian III-598
Chen, Xingxing III-3
Chen, Xingyu II-28
Chen, Xue I-244
Chen, Yahong II-406
Chen, Yanbing I-76
Chen, Yang II-255
Chen, Yingwen I-314, I-401, II-206
- Chen, Zekai III-393
Chen, Zhen I-165
Chen, Zihan II-548
Cheng, Guang II-548, III-96
Cheng, Jiujun I-175
Cheng, Xiaolu I-52
Cheng, Xiuzhen II-609
Cheng, Yuyang II-356
Cheng, Zesheng I-101
Chu, Zihao III-507
Cong, Peijin III-380
Cui, Jie III-249
Cui, Lei III-507
Cui, Meng III-481, III-572
Cui, Ningning I-271
Cui, Xuande I-3
Cui, Yue I-591
Cui, Zhenglong II-291
- Da Teng, I-296
Dai, Haipeng II-317
Dang, Na I-526
Dang, Xiaochao I-216, II-267
Deng, Boyu II-417
Deng, Haojiang II-384
Di, Xiaoqiang I-153, II-489
Ding, Jiaxu I-364
Ding, Shuai II-384
Ding, Xu I-196, III-495
Ding, Youwei III-333
Ding, Zhaoyun II-539, III-657
Dong, Anming I-351, II-3
Dong, Jing II-193
Dong, Xinhua III-3
Du, Gewangzi I-271
Du, Jiarong III-261
Du, Ruizhong I-567
Duan, Xincheng I-351
- Fan, Wanshu I-15
Fan, Xin III-431
Fan, Yuqi III-598
Fang, Chen III-367, III-495

- Fang, Jun I-185
 Fang, LiMing I-427
 Fang, Liming II-93
 Fang, Luyue II-489
 Farea, Ebraheem III-563
 Feng, Jiaqi II-243
 Feng, Shuai I-175
 Feng, Tao II-557
 Feng, Xiaotao I-466
 Feng, Xue-cai I-326
 Feng, Yue II-582
 Fu, Hao I-466
 Fu, Lei II-609
 Fu, Lianyou I-231
 Fu, Nan II-548
 fu, Shaojing I-401
- Gan, Jianyuan III-617
 Gao, Caina I-643
 Gao, Guoju III-178
 Gao, Jie I-139
 Gao, Qinghe III-638
 Gao, Ruipeng I-185, II-155
 Gao, Shang I-244
 Gao, Shouwan II-105, III-213
 Gao, Xianming II-557
 Gao, Yongqiang III-607
 Gao, Zhipeng II-181
 Ge, Jingguo II-384
 Geng, Jingru II-64
 Gu, Chen I-3
 Gu, Chengjie III-249
 Gu, Qi III-586
 Guo, Dongyu III-31
 Guo, Feng III-84
 Guo, Linlin I-643
 Guo, Qing II-218
 Guo, Shaoyong III-456
 Guo, Xin III-117
 Guo, Xing III-72
 Guo, Ying III-238
 Guo, Yonghe III-507
- Han, Chengzhuo II-279
 Han, Hongmu III-3
 Han, Yubing I-364
 Han, Zhenhua II-231
 Hao, Qixia III-273
 Hao, Zhanjun I-216, II-267
- Hao, Zhiyu III-507
 He, Guofeng II-644
 He, Hang III-481
 He, Qian III-419
 He, Shuang II-609
 He, Sihan II-93
 He, Xiaoming III-47
 He, Xin III-59, III-72
 He, Yunhua I-244, II-356
 He, Zhijie II-393, II-595
 Hou, Kaixiang I-88
 Hou, Xiaowei II-501
 Hsieh, Chao-Hsien II-16
 Hu, Cheng I-545
 Hu, Chunqiang I-500, II-514
 Hu, Di I-27
 Hu, Donghui I-3
 Hu, Haibo II-514
 Hu, Huanfeng III-354
 Hu, Pengfei I-466, II-609
 Hu, Shiyan III-380
 Hu, Zhaolong I-113
 Huang, Baoqi I-615
 Huang, Chao I-296
 Huang, Fangzheng II-343
 Huang, He III-178
 Huang, Jianxin III-47
 Huang, Jintao I-603
 Huang, Min III-321
 Huang, Ning I-326, II-375
 Huang, Songping II-539
 Huang, Weiqing II-64, III-84
 Huang, Xuefei II-450
- Ji, Hui I-643
 Ji, Minghao II-393, II-595
 Jiang, Bingcheng III-419
 Jiang, Di II-514
 Jiang, Han I-139
 Jiang, Lin II-303
 Jiang, Shan II-155
 Jiang, Shang II-582
 Jiang, Shanqing II-557
 Jiang, Weiming III-380
 Jiang, Xufeng I-283
 Jiang, Xuxiang III-127
 Jin, Yan I-377
 Jing, Tao III-638
 Ju, Yanli III-468

- Kang, Hui I-557
 Ke, Wei I-76, II-450
 Kong, Qingshan III-84

 Lai, Ruilin III-285
 Lei, Jianjun III-345
 Lei, Yan II-514
 Li, Bo II-489
 Li, Changfeng II-16
 Li, Chao II-343
 Li, Fan III-431
 Li, Fenghua II-406
 Li, Fengqi I-579
 Li, Fengyin II-475
 Li, Guangshun I-40
 Li, Hong I-64, I-603
 Li, Hongjuan I-557
 Li, Jiahui I-557
 Li, Jianbo I-101, I-206, II-168, II-441,
 III-629
 Li, Ke I-64
 Li, Keqiu I-413, III-155
 Li, Lu I-283
 Li, Mingchu III-563
 Li, Minghao III-285
 Li, Minglu I-113
 Li, Mingxia II-231
 Li, Mingxuan I-338
 Li, Mohan I-258
 Li, Peichen III-321
 Li, Ruihao III-202
 Li, Sufang II-3
 Li, Tailai III-296
 Li, Wenjing III-456
 Li, Xiaofan II-255
 Li, Xiaohu II-330
 Li, XinRan III-648
 Li, Xiong III-529
 Li, Yanbin I-389, II-93
 Li, Ying III-629
 Li, Yixin II-330
 Li, Zhaowei I-526
 Liang, Binbin III-72
 Liang, Jian II-393, II-595
 Liao, Duoyue III-3
 Lin, Aixian II-572
 Lin, Bin II-40, III-541
 Lin, Chunxin III-629
 Lin, Feilong I-113
 Lin, Guiping III-59
 Lin, Yaguang III-261
 Liu, Chunfeng III-225
 Liu, Dong III-481, III-572
 Liu, Dunge II-255
 Liu, Gaoyuan I-216
 Liu, Jia II-28
 Liu, Jie III-139
 Liu, Liang III-333
 Liu, Lin II-40
 Liu, Long II-130
 Liu, Peipei III-139
 Liu, Peng III-419
 Liu, Rui II-193
 Liu, Runrong I-231
 Liu, Shui I-185
 Liu, Tang II-117
 Liu, Xiaowu I-526, III-108
 Liu, Xinyu II-460
 Liu, Xiuli I-15
 Liu, Xiulong I-413
 Liu, Xu II-193, II-489
 Liu, Xuan III-431
 Liu, Yan I-567
 Liu, Yanbing I-545
 Liu, Yao II-80, III-225
 Liu, Yiqing II-64
 Liu, Yongji III-507
 Liu, Zewei I-500
 Liu, Zhe I-389, II-93
 Liu, Zhen I-139
 Liu, Zhongjin I-603
 Lu, Bingxian II-365
 Lu, Bo I-175
 Lu, Qing II-644
 Lu, Wanying III-333
 Lu, Wei I-185
 Lu, Xiaozhen III-117
 Lu, Yang I-27, I-165
 Luo, Hui I-27, II-526
 Luo, Juan III-431
 Luo, Qi III-393
 Luo, Sinian I-389
 Luo, Yuchuan I-314, I-401
 Lv, Qiuqian II-501
 Lv, Shichao I-603
 Lv, Yang I-101
 Lv, Zefang I-441

- Lv, Zhiqiang I-101, I-206, II-168, III-84
 Lyu, Zengwei III-598
 lyu, ZengWei III-648
- Ma, Chuan I-427
 Ma, Haoran III-657
 Ma, Huadong I-126
 Ma, Nan I-185
 Ma, Wenshuo I-526, III-108
 Ma, Xiu II-501
 Ma, Xuebin II-572
 Ma, Zhaobin II-168
 Ma, Zhenxian I-627
 Mao, Qichao I-175
 Mao, Yingchi III-47
 Meng, Chuize II-155
 Meng, Kelong II-142
 Meng, Lili I-643
 Meng, Xianglong II-441
 Mo, Zijia II-181
 Mou, Wenjie III-202
- Nawaf, Liqaa III-617
 Ni, Shenggang III-31
 Nian, Aixin II-539
 Ning, Kaiwen III-165
 Niu, Ben II-406
 Niu, Jianwei III-481, III-572
 Niu, Longsheng III-238
 Niu, Qiang II-105, III-213
 Niu, ShuoShuo III-657
- Ouyang, Fang I-113
- Pan, Jinhao I-579
 Pang, Deming II-206
 Peng, Jian II-117, II-218
 Peng, Jianfei III-333
 Ping, Ping III-47
- Qi, Shengyuan II-557
 Qi, Shuang III-541
 Qian, Liping III-541
 Qian, Pengcheng III-261
 Qian, Xiang-yun II-375
 Qian, Yuhang II-609
 Qiao, Ying III-431
 Qin, Yiping II-489
 Qiu, Fengyuan I-427
- Qiu, Jing II-3
 Qiu, Qiqi I-231
 Qiu, Sen II-130
 Qiu, Tie I-88, III-155, III-273
 Qiu, Xuesong III-456
 Qiu, Zhaohua II-52, II-64
 Qu, Guanqun I-258
 Qu, Wenyu III-225
 Qu, Zhihao II-80
- Rana, Omer F. III-617
 Ren, Tao III-481, III-572
- Shah, Syed Bilal Hussain III-617
 Shen, Bo III-586
 Sheng, Hao I-76, II-291, II-450
 Sheng, Zhaoyu II-168
 Shi, Gang I-271
 Shi, Lei III-367, III-495, III-598
 Shi, Shuyu II-317
 Shi, Yi III-367, III-495
 Shi, Zhiqiang I-603
 Shi, Zhixin I-513
 Shu, Jian III-354
 Shu, Xiaowei III-165
 Sima, Qinghua III-178
 Song, Chenliu III-629
 Song, Jing III-419
 Song, Jinpeng II-317
 Song, Yubo II-16
 Sun, Degang I-591, II-582
 Sun, Fuyong I-185
 Sun, Geng I-557
 Sun, Guangling I-165
 Sun, Han I-338
 Sun, Haokai II-168
 Sun, Haoran I-453
 Sun, Jiabao II-460
 Sun, Jiande I-643
 Sun, Juping I-153
 Sun, Limin I-64, I-603, III-139
 Sun, Mengjie I-64
 Sun, Weifeng II-142
 Sun, Yanbin I-258
 Sun, Yu-E III-178
 Sun, Yuhong II-475
- Tan, Guoping III-309
 Tan, Haisheng II-231

- Tan, Jianlong I-545
Tang, Bin II-80, III-444
Tang, Changbing I-113
Tang, Qi I-615
Tao, Dan II-155
Tao, Xiaoling I-231
Tian, Xiang I-52
Tong, Xinyu I-413
- Wan, Haoran II-317
Wang, Ailian II-142
Wang, Baolin I-500
Wang, Dan III-345
Wang, Deqing II-621
Wang, Dongsheng I-314
Wang, Fang II-417
Wang, Fei II-539
Wang, Guijuan I-364
Wang, Hai II-303
Wang, Haitao III-3
Wang, Hao III-165
Wang, Haowei I-153
Wang, Haoyu I-126
Wang, Hua II-475
Wang, Huan II-609
Wang, Huihui II-130
Wang, Jiale III-202
Wang, Jian I-557
Wang, Jianrong I-453
Wang, Jinfan III-139
Wang, Jingchao II-417
Wang, Jinhua III-657
Wang, Jun III-127
Wang, Kang III-507
Wang, Lei II-317
Wang, Liang III-261
Wang, Lin II-621
Wang, Mingqiang I-491, I-535
Wang, Nan II-460
Wang, Peng II-330
Wang, Pingchuan I-579
Wang, Qian II-181
Wang, Ran I-627, III-406
Wang, Shicheng III-419
Wang, Shiyu II-475
Wang, Shuai II-303
Wang, Siye I-591, II-582
Wang, Tianpeng III-345
Wang, Wei II-365, III-155
Wang, Wen II-52, II-64
- Wang, Wenyue I-52
Wang, Xia II-28
Wang, Xiao II-393, II-595
Wang, Xiaofei III-468
Wang, Xiaolei I-535
Wang, Xin III-468
Wang, Xingwei III-321
Wang, Xinying III-468
Wang, Xiuxiu II-526
Wang, Xuan II-28
Wang, Yan II-501
Wang, Yang I-535
Wang, Yihuai III-178
Wang, Yingdong I-296
Wang, Yizong I-126
Wang, Yue II-441
Wang, Yuejiao II-267
Wang, Yunhai II-393, II-595
Wang, Zhelong II-130
Wang, Zhen II-16
Wang, Zhifei I-615
Wang, Zhigang III-345
Wang, Zhou II-343
Wei, Da I-557
Wei, Dong II-52
Wei, Lu III-249
Wei, Xi I-139
Wei, Xing I-27, I-165, II-526
Wei, Yuting II-206
Wei, Zhenchun III-598
Wei, ZhenChun III-648
Wei, Zijun II-548
Wu, Di II-526
Wu, Fan III-59
Wu, Guobin II-393, II-595
Wu, Guoliang III-468
Wu, Guowei III-165
Wu, Hao I-513
Wu, Jun III-47
Wu, Junhua I-40
Wu, Lin III-554
Wu, Mengning II-155
Wu, Mingli II-429
Wu, Tongshuai I-271
Wu, Wei III-309
Wu, Weibin I-389, II-93
Wu, Yuan III-541
Wu, Yuchen III-225
Wu, Yueqing I-466

- Xi, Heran III-191
 Xia, Hui I-326, II-375
 Xia, Rui II-417
 Xiang, Zhengzhong II-117, II-218
 Xiangli, Peng III-529
 Xiao, Fu II-317
 Xiao, Ke II-356
 Xiao, Liang I-441
 Xiao, Xuan II-155
 Xiao, Yin hao III-127
 Xie, Zaipeng II-80
 Xing, Mengyan I-513
 Xing, Weiwei I-185
 Xiong, Hu II-644
 Xiong, Yonghong I-175
 Xiong, Zhang I-76
 Xu, Chao I-88
 Xu, Chengxin III-108
 Xu, Gang I-615
 Xu, Guangliao III-202
 Xu, Hao II-384
 Xu, Jing III-367, III-495
 Xu, Jingxiang III-238
 Xu, Juan III-367, III-648
 Xu, Lidong I-491
 Xu, Ming I-401
 Xu, Minghao II-557
 Xu, Minghui I-466, II-609
 Xu, Pengfei I-196, II-393, II-595
 Xu, Rui I-153
 Xu, Shiyu I-441
 Xu, Shiyuan I-244, I-479, II-356
 Xu, Sicong III-59
 Xu, Tianyi I-88, I-453
 Xu, Weikai II-621
 Xu, WenZheng II-218
 Xu, Xin II-635
 Xu, Yifei II-384
 Xu, Zhen I-338
 Xu, Zheng III-607
 Xu, Zhigang III-3
 Xu, Zhihao II-168
 Xu, Zhou III-165
 Xu, Ziheng II-548
 Xue, Guangtao I-314, II-206
 Xun, Kai II-28

 Yan, Biwei I-52, I-351, I-364
 Yan, Hao III-444
 Yan, Longchuan III-507

 Yan, Yijie I-479
 Yan, Zhongzhen III-3
 Yang, Da II-291
 Yang, Fan III-249
 Yang, Fuyu I-126
 Yang, Guoming III-456
 Yang, Hao I-427
 Yang, Jiayi III-529
 Yang, Mengqing I-377
 Yang, Mingchuan III-191
 Yang, Panlong III-59
 Yang, Tingting II-279
 Yang, Xiu-gui I-326, II-375
 Yang, Xu I-401, II-105, III-213
 Yang, Yue III-191
 Yang, Zhaoxian II-343
 Yang, Zhizheng II-317
 Yang, Zhongyuan II-557
 Yao, Shang I-27
 Yao, Yan I-364
 Yao, Yanqing I-296
 Yao, Yiming III-572
 Ye, Baoliu III-444
 Ye, Chunming I-377
 Ye, Chunxiao I-377
 Ye, Jiahui III-202
 Ye, Kangze III-3
 Ye, Lingling II-231
 Ye, Rongkun I-206
 Yi, Bo III-321
 Yi, Changyan I-627, II-243, III-20, III-406
 Yi, Pengfei II-193
 Yin, Guangqiang II-644
 Yin, Luxiu III-431
 Yin, Yuqing II-105, III-213
 Ying, Xiang I-139
 You, Minghang II-621
 Yu, Hao I-165
 Yu, Jiguo I-52, I-351, I-364, II-3
 Yu, Jinxin III-20
 Yu, Kan I-40
 Yu, Shi I-441
 Yu, Xiaojie II-105, III-213
 Yu, Xinlei II-181
 Yu, Yuelin I-231
 Yu, Ziqiang II-393, II-595
 Yuan, Fangfang I-545
 Yuan, Genji II-441
 Yuan, Guiyuan I-175
 Yuan, Xiaohui III-598

- Yuan, XiaoHui III-648
 Yuan, Yuan I-52
 Yuen, Tsz Hon II-429

 Zan, Fengbiao I-88
 Zeng, Feng III-554
 Zeng, Jiixin III-273
 Zhai, Wenbin III-333
 Zhai, Yan I-165
 Zhang, Bixun I-196
 Zhang, Chaokun III-296
 Zhang, Chaoyue II-40, III-541
 Zhang, Chi II-231
 Zhang, Chuanting II-3
 Zhang, Chunling III-345
 Zhang, Chunyan I-545
 Zhang, Daiyang I-216, II-267
 Zhang, Deyong III-321
 Zhang, Gongxuan III-380
 Zhang, Guoming I-466, II-609
 Zhang, Haijing I-231
 Zhang, Hongniu I-258
 Zhang, Huanle II-609
 Zhang, Huiru I-40
 Zhang, Jia I-643
 Zhang, Jing I-513, III-249
 Zhang, Jingjing II-635
 Zhang, Jiuwu I-413
 Zhang, Junyi II-330
 Zhang, Kehan II-489
 Zhang, Li I-351, I-364, II-3, II-526
 Zhang, Lingyu II-393, II-595
 Zhang, Lupeng I-579
 Zhang, Mingxin III-638
 Zhang, Ning III-84
 Zhang, Penghui II-393, II-595
 Zhang, Qihang I-413
 Zhang, Ran II-40
 Zhang, Rui I-326, II-375, III-238
 Zhang, Ruixuan I-139
 Zhang, Runfa III-563
 Zhang, Ruyun I-389, III-117
 Zhang, Songwei III-155
 Zhang, Teng II-3
 Zhang, Tong II-243, III-20
 Zhang, Wei III-127
 Zhang, Weidong I-64
 Zhang, Wenqiu III-406
 Zhang, Xiao III-393
 Zhang, Xiaoling I-603
 Zhang, Xiaosong III-529
 Zhang, Yan I-338
 Zhang, Yanfang I-591, II-582
 Zhang, Yang II-365
 Zhang, Yaoxue I-185
 Zhang, Ying II-393, II-595
 Zhang, ZeYu III-648
 Zhang, Zhaogong II-635
 Zhang, Zheng III-554
 Zhao, Aite I-206
 Zhao, Bin I-466
 Zhao, Chen II-181
 Zhao, Chong I-27, I-165, II-526
 Zhao, Deyu III-96
 Zhao, Dong I-126
 Zhao, Gansen III-285
 Zhao, Hongjian III-354
 Zhao, Hongkai II-130
 Zhao, Jiapeng I-603
 Zhao, Weiyi II-80
 Zhao, Xunwei III-345
 Zhao, Yu II-539
 Zhao, Yubin II-255
 Zhao, Yuxin II-291
 Zhao, Yuyu III-96
 Zhao, Zhao III-225
 Zhao, Zhihong II-28
 Zhen, Lin I-567
 Zheng, Hang I-196
 Zheng, Xiang I-196
 Zheng, Xiao III-617
 Zheng, Yanwei III-393
 Zheng, Yaowen I-64
 Zheng, Yuqi II-330
 Zhong, Fangtian III-393
 Zhong, Hong III-249
 Zhou, Chunyue III-638
 Zhou, Dongsheng I-15, II-193, II-539,
 III-657
 Zhou, Junlong III-380
 Zhou, Lei I-296
 Zhou, Lixiao I-113
 Zhou, Lu I-427, II-93, III-117
 Zhou, Siyuan III-309
 Zhou, Xiaobo I-88, III-273, III-296
 Zhou, Xiaolei II-303
 Zhou, You II-3
 Zhou, Yubin III-31
 Zhou, Yuyang II-548
 Zhou, Zejun II-406

Zhu, Chenguang I-271
Zhu, Dejun I-153
Zhu, Fang III-20
Zhu, Hongsong III-139
Zhu, Jinghua III-191
Zhu, Kun I-627, III-406

Zhu, Ruixing III-96
Zhu, Shilin II-303
Zhu, Yuanyuan III-617
Zhu, Yuchen I-153
Zou, Qian III-84
Zou, Yifei I-466