

Yijia Chen
Xiaotie Deng
Mei Lu (Eds.)

LNCS 11458

Frontiers in Algorithmics

13th International Workshop, FAW 2019
Sanya, China, April 29 – May 3, 2019
Proceedings

 Springer

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board Members

David Hutchison

Lancaster University, Lancaster, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Friedemann Mattern

ETH Zurich, Zurich, Switzerland

John C. Mitchell

Stanford University, Stanford, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

C. Pandu Rangan

Indian Institute of Technology Madras, Chennai, India

Bernhard Steffen

TU Dortmund University, Dortmund, Germany

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

More information about this series at <http://www.springer.com/series/7407>

Yijia Chen · Xiaotie Deng ·
Mei Lu (Eds.)

Frontiers in Algorithmics

13th International Workshop, FAW 2019
Sanya, China, April 29 – May 3, 2019
Proceedings

Editors

Yijia Chen
Fudan University
Shanghai, China

Xiaotie Deng
Peking University
Beijing, China

Mei Lu
Tsinghua University
Beijing, China

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-030-18125-3 ISBN 978-3-030-18126-0 (eBook)
<https://doi.org/10.1007/978-3-030-18126-0>

LNCS Sublibrary: SL1 – Theoretical Computer Science and General Issues

© Springer Nature Switzerland AG 2019

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

This volume contains the papers presented at FAW 2019: the 13th International Frontiers of Algorithmics Workshop, held during April 29 – May 3, 2019, at the Tsinghua Sanya International Mathematics Forum, in Sanya, P. R. China. The workshop provides a focused forum on current trends in research on algorithms, discrete structures, and their applications, and brings together international experts at the research frontiers in these areas to exchange ideas and to present significant new results.

The Program Committee, consisting of 28 top researchers from the field, reviewed 21 submissions and decided to accept 15 papers. Each paper had at least three reviews, with additional reviews solicited as needed. The review process was conducted entirely electronically via EasyChair. We are grateful to the EasyChair team for allowing us to handle the submissions and the review process and to the Program Committee for their insightful reviews and discussions, which made our job easier.

Besides the regular talks, the program also included three keynote talks by Yuqing Kong (Peking University), Bundit Laekhanukit (Shanghai University of Finance and Economics), and Jian Li (Tsinghua University).

We are very grateful to all the people who made this meeting possible: the authors for submitting their papers, the Program Committee members and external reviewers for their excellent work, and the three keynote speakers.

In particular, we would like to thank the Tsinghua Sanya International Mathematics Forum (TSIMF) for hosting the conference and providing financial and organizational support. In addition to the ordinary FAW conference talks, 20 participating Program Committee members gave talks, organized in several sessions and research mini-workshops on current frontiers of computer science theory and mathematical foundations. In addition, FAW organized several special topic research sessions on approximate algorithms, algorithmic game theory, and parameterized complexity, which were aligned with the keynote speeches. TSIMF allowed FAW to organize a rich event to bring together more than 50 active researchers for academic exchanges on the current progress in algorithms and complexity.

Finally, we would like to thank the members of the Editorial Board of *Lecture Notes in Computer Science* and the editors at Springer for their encouragement and cooperation throughout the preparation of this conference.

March 2019

Yijia Chen
Xiaotie Deng
Mei Lu

Organization

Program Committee Chairs

Yijia Chen	Fudan University, China
Xiaotie Deng	Peking University, China
Mei Lu	Tsinghua University, China

Steering Committee

Jianer Chen	Texas A&M University, USA
Xiaotie Deng	Peking University, China
John E. Hopcroft	Cornell University, USA
Wei Li	Beihang University, China
Huimin Lin	Chinese Academy of Sciences, China
Andrew Chi-Chih Yao	Tsinghua University, China
Binhai Zhu	Montana State University, USA

Program Committee

Jin-Yi Cai	University of Wisconsin - Madison, USA
Yixin Cao	Hong Kong Polytechnic University, SAR China
Xujin Chen	Chinese Academy of Sciences, China
Yukun Cheng	Suzhou University of Science and Technology, China
Hu Ding	University of Science and Technology of China, China
Ran Duan	Tsinghua University, China
Mordecai J. Golin	Hong Kong University of Science and Technology, SAR China
Heng Guo	University of Edinburgh, UK
Zhiyi Huang	University of Hong Kong, SAR China
Kazuo Iwama	Kyoto University, Japan
Jian Li	Tsinghua University, China
Shimin Li	Winona State University, USA
Minming Li	City University of Hong Kong, SAR China
Bingkai Lin	National Institute of Informatics, Japan
Tian Liu	Peking University, China
Pinyan Lu	Shanghai University of Finance and Economics, China
Pan Peng	University of Sheffield, UK
Richard Peng	Georgia Institute of Technology, USA
Qi Qi	Hong Kong University of Science and Technology, SAR China
Dominik Scheder	Shanghai Jiao Tong University, China
Haitao Wang	Utah State University, USA

Mingji Xia	Chinese Academy of Sciences, China
Deshi Ye	Zhejiang University, China
Yang Yuan	MIT, USA
Binhai Zhu	Montana State University, USA

Additional Reviewers

Chenglin Fan	Eiji Miyano
Yong Gu	Cong Wang
Yaonan Jin	Wenwei Wang
Ning Kang	Jie Xue
Bundit Laekhanukit	Chenhao Wang
Weian Li	Chenchen Wu
Xianyue Li	Jingru Zhang
Jinyan Liu	Peng Zhang

Contents

A Polynomial Time Algorithm for Fair Resource Allocation in Resource Exchange	1
<i>Xiang Yan and Wei Zhu</i>	
A Local Search $4/3$ -approximation Algorithm for the Minimum 3-path Partition Problem	14
<i>Yong Chen, Randy Goebel, Guohui Lin, Longcheng Liu, Bing Su, Weitian Tong, Yao Xu, and An Zhang</i>	
Efficient Guarding of Polygons and Terrains	26
<i>Pradeesha Ashok and Meghana M. Reddy</i>	
Graph Orientation with Edge Modifications	38
<i>Yuichi Asahiro, Jesper Jansson, Eiji Miyano, Hirotaka Ono, and Sandhya T. P.</i>	
Local Coloring: New Observations and New Reductions	51
<i>Jie You, Yixin Cao, and Jianxin Wang</i>	
Secure Computation of Any Boolean Function Based on Any Deck of Cards	63
<i>Kazumasa Shinagawa and Takaaki Mizuki</i>	
A Novel Business Model for Electric Car Sharing.	76
<i>Yukun Cheng, Xiaotie Deng, and Mengqian Zhang</i>	
Constructing Three Completely Independent Spanning Trees in Locally Twisted Cubes	88
<i>Kung-Jui Pai, Ruay-Shiung Chang, Jou-Ming Chang, and Ro-Yu Wu</i>	
Read-Once Resolutions in Horn Formulas	100
<i>Hans Kleine Büning, P. Wojciechowski, and K. Subramani</i>	
Vertex-Critical (P_5 , banner)-Free Graphs	111
<i>Qingqiong Cai, Shenwei Huang, Tao Li, and Yongtang Shi</i>	
An FPTAS for Stochastic Unbounded Min-Knapsack Problem	121
<i>Zhihao Jiang and Haoyu Zhao</i>	
The Inapproximability of k -Dominating Set for Parameterized AC^0 Circuits	133
<i>Wenxing Lai</i>	

Mutual Visibility by Robots with Persistent Memory 144
Subhash Bhagat and Krishnendu Mukhopadhyaya

**Pushing the Online Matrix-Vector Conjecture Off-Line and Identifying
Its Easy Cases** 156
*Leszek Gąsieniec, Jesper Jansson, Christos Levcopoulos,
Andrzej Lingas, and Mia Persson*

**An Improved Approximation Algorithm for the k -Means Problem
with Penalties** 170
Qilong Feng, Zhen Zhang, Feng Shi, and Jianxin Wang

Author Index 183



A Polynomial Time Algorithm for Fair Resource Allocation in Resource Exchange

Xiang Yan^{1(✉)} and Wei Zhu²

¹ Shanghai Jiao Tong University, Shanghai 200240, China
xyansjtu@163.com

² China Academy of Aerospace Standardization and Product Assurance,
Beijing 100071, China
shakerswei@sina.com

Abstract. The rapid growth of wireless and mobile Internet has led to wide applications of exchanging resources over network, in which how to fairly allocate resources has become a critical challenge. To motivate sharing, a *BD Mechanism* is proposed for resource allocation, which is based on a combinatorial structure called *bottleneck decomposition*. The mechanism has been shown with properties of fairness, economic efficiency [17], and truthfulness against two kinds of strategic behaviors [2,3]. Unfortunately, the crux on how to compute a bottleneck decomposition of any graph is remain untouched. In this paper, we focus on the computation of bottleneck decomposition to fill the blanks and prove that the bottleneck decomposition of a network $G = (V, E; w_v)$ can be computed in $O(n^6 \log(nU))$, where $n = |V|$ and $U = \max_{v \in V} w_v$. Based on the bottleneck decomposition, a fair allocation in resource exchange system can be obtained in polynomial time. In addition, our work completes the computation of a market equilibrium and its relationship to two concepts of fairness in resource exchange.

Keywords: Polynomial algorithm · Fair allocation · Resource exchange · Bottleneck decomposition

1 Introduction

The Internet era has witnessed plenty of implementations of resource exchange [10–14]. It embodies the essence of the *sharing economy* and captures the ideas of collaborative consumption of resource (such as the bandwidth) to networks with participants (or agents) [7], such that agents can benefit from exchanging each own idle resource with others. In this paper, we study the resource exchange problem over networks, which goes beyond the peer-to-peer (P2P) bandwidth

Supported by the National Nature Science Foundation of China (No. 11301475, 61632017, 61761146005).

sharing idea [17]. Peers in such networks act as both suppliers and customers of resources, and make their resources directly available to other network peers according to preset network rules [16].

The resource exchange problem can be formally modeled on an undirected connected graph $G = (V, E; w)$, where each vertex $u \in V$ represents an agent with w_u units of divisible idle resources (or weight) to be distributed among its neighbor set $\Gamma(u)$. The utility U_u is determined by the total amount of the resources obtained from its neighbors. Define x_{uv} to be the fraction of resource that agent u allocates to v and call the collection $X = (x_{uv})$ an *allocation*. Then the utility of agent u under allocation X is $U_u = \sum_{v \in \Gamma(u)} x_{vu} w_v$, subject to the constraint of $\sum_{v \in \Gamma(u)} x_{uv} \leq 1$.

One critical issues for the resource exchange problem is how to design a resource exchange protocol to maintain agents' participation in a fair fashion. Ideally, the resource each agent obtains can compensate its contribution, but such a state may not exist due to the structure of the underlying networks. Thus Georgiadis *et al.* [9] thought that an allocation is fair if it can balance the exchange among all agents as much as possible. An *exchange ratio* of each agent is defined then, to quantify the utility it receives per unit of resource it delivers out, i.e. $\beta_u(X) = \frac{U_u}{\sum_{v \in \Gamma(u)} x_{uv} w_v}$ for given allocation X . In [9], an allocation X is said to be fair, if its exchange ratio vector $\beta(X) = (\beta_u(X))_{u \in V}$ is lexicographic optimal (lex-optimal for short). And a polynomial-time algorithm is designed to find such a fair allocation by transforming it to a linear programming problem.

On the other hand, Wu and Zhang [17] pioneered the concept of ‘‘proportional response’’ inspired by the idea of ‘‘tit-for-tat’’ for the consideration of fairness. Under a proportional response allocation, each agent responds to the neighbors who offer resource to it by allocating its resource in proportion to how much it receives. Formally, the proportional response allocation is specified by $x_{uv} = \frac{x_{vu} w_v}{\sum_{k \in \Gamma(u)} x_{ku} w_k}$. The authors showed the equivalence between such a fair allocation and the *market equilibrium* of a pure exchange economy in which each agent sells its own resource and uses the money earned through trading to buy its neighbors' resource.

The algorithm to get a fair allocation in [17] includes two parts: computing a combinatorial decomposition, called *bottleneck decomposition*, of a given graph; and constructing a market equilibrium from the bottleneck decomposition. The work in [17] only involved the latter. But how to compute the bottleneck decomposition is remained untouched. In this paper, we design a polynomial time algorithm to address the computation of the bottleneck decomposition. In addition, we show the equilibrium allocation (i.e. the allocation of the market equilibrium) from the bottleneck decomposition also is lex-optimal. Such a result establishes the connection between the two concepts of fairness in [9] and [17].

Another contribution of this work is to complete the computation of the market equilibrium in a special setting of a linear exchange market. The study of market equilibrium has a long and distinguished history in economics, starting with Arrow and Deberu's solution [1] which proves the existence of the market equilibrium under mild conditions. Much work has focused on the computational

aspects of the market equilibrium. Specially for the linear exchange model, Eaves [5] first presented an exact algorithm by reducing to a linear complementary problem. Then Garg *et al.* [8] derived the first polynomial time algorithm through a combinatorial interpretation and based on the characterization of equilibria as the solution set of a convex program. Later Ye [18] showed that a market equilibrium can be computed in $O(n^8 \log^2 U)$ -time with the interior point method. Recently, Duan *et al.* [4] improved the running time to $O(n^7 \log^2(nU))$ by a combinatorial algorithm. Compared with the general linear exchange model, we further assume that the resource of any agent is treated with equal preference by all his neighbors. Based on it, Wu and Zhang [17] proposed the bottleneck decomposition, which decomposes participants in the market into several components, and showed that in a market equilibrium, trading only happens within each component. Therefore, the problem of computing a market equilibrium is reduced to one of computing the bottleneck decomposition. Our main task in this paper is to design a polynomial time algorithm to compute the bottleneck decomposition and to complete the computation of a market equilibrium in [17]. The time complexity of our algorithm is $O(n^6 \log(nU))$, which is better than the algorithm of Duan *et al.* [4], because of further assumption in our setting.

In the rest of this paper, we introduce the concepts and properties of bottleneck decomposition in Sect. 2 and provide a polynomial time algorithm for computing the bottleneck decomposition in Sect. 3. In Sect. 4 we describe a market equilibrium from the bottleneck decomposition and show the fairness of the equilibrium allocation. At last we conclude this paper in Sect. 5.

2 Preliminary

We consider a resource exchange problem modeled on an undirected and connected graph $G = (V, E; w)$ with vertex set V and edge set E , respectively, and $w : V \rightarrow R^+$ is the weight function on vertex set. Let $\Gamma(i) = \{j : (i, j) \in E\}$ be the set of vertices adjacent to i in G , i.e. the neighborhood of vertex i . For each vertex subset $S \subseteq V$, define $w(S) = \sum_{i \in S} w_i$ and $\Gamma(S) = \cup_{i \in S} \Gamma(i)$. Note that it is possible $S \cap \Gamma(S) \neq \emptyset$ and if $S \cap \Gamma(S) = \emptyset$, then S must be independent. For each S , define $\alpha(S) = \frac{w(\Gamma(S))}{w(S)}$, referred to as the inclusive expansion ratio of S , or the α -ratio of S for short. It is not hard to observe that the neighborhood $\Gamma(V)$ is still V and its α -ratio is $\alpha(V) = 1$.

Definition 1 (Bottleneck and Maximal Bottleneck). *A vertex subset $B \subseteq V$ is called a bottleneck of G if $\alpha(B) = \min_{S \subseteq V} \alpha(S)$. If bottleneck B is a maximal bottleneck, then for any subset \tilde{B} with $B \subset \tilde{B} \subseteq V$, it must be $\alpha(\tilde{B}) > \alpha(B)$. We name $(B, \Gamma(B))$ as the maximal bottleneck pair of G .*

From Definition 1, we can understand the maximal bottleneck as the bottleneck whose size is maximal. Wu and Zhang [17] showed that the maximal bottleneck of any graph is unique and proposed the following bottleneck decomposition with the help of the uniqueness.

Definition 2 (Bottleneck Decomposition). Given an undirected and connected graph $G = (V, E; w)$. Start with $V_1 = V$, $G_1 = G$ and $i = 1$. Find the maximal bottleneck B_i of G_i and let G_{i+1} be the induced subgraph on the vertex set $V_{i+1} = V_i - (B_i \cup C_i)$, where $C_i = \Gamma(B_i) \cap V_i$, the neighbor set of B_i in the subgraph G_i . Repeat if $G_{i+1} \neq \emptyset$ and set $k = i$ if $G_{i+1} = \emptyset$. Then we call $\mathcal{B} = \{(B_1, C_1), \dots, (B_k, C_k)\}$ the bottleneck decomposition of G , $\alpha_i = \frac{w(C_i)}{w(B_i)}$ the i -th α -ratio and $(\alpha_i)_{i=1}^k$ the α -ratio vector.

We propose an example in the following to show the bottleneck decomposition of a graph. Consider the graph of Fig. 1 which has 6 vertices. The numbers in each circle represents the weight of each vertex. At the first step, $G_1 = G$, $V_1 = V$ and the maximal bottleneck pair of G_1 is $(B_1, C_1) = (\{v_1, v_2\}, \{v_3, v_4\})$ with $\alpha_1 = \frac{1}{2}$. After removing $B_1 \cup C_1$, $V_2 = \{v_5, v_6\}$, $G_2 = G[V_2]$ and the maximal bottleneck pair of G_2 is $(B_2, C_2) = (\{v_5, v_6\}, \{v_5, v_6\})$ with $\alpha_2 = 1$. Therefore the bottleneck decomposition is $\mathcal{B} = \{(\{v_1, v_2\}, \{v_3, v_4\}), (\{v_5, v_6\}, \{v_5, v_6\})\}$.

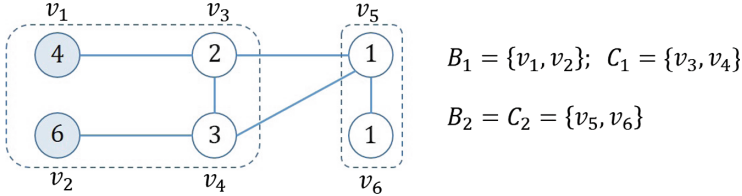


Fig. 1. The figure illustrates bottleneck decomposition of a network.

The bottleneck decomposition has some combinatorial properties which are very crucial to the study on the truthfulness of BD Mechanism in [2] and [3] and to the discussion of the fairness of resource allocation. Although Wu and Zhang mentioned these properties in [17], their proofs are not included. We explain these properties here and present the proofs in full paper.

Proposition 1. Given an undirected and connected graph $G = (V, E; w)$, the bottleneck decomposition \mathcal{B} of G satisfies

- (1) $0 < \alpha_1 < \alpha_2 < \dots < \alpha_k \leq 1$;
- (2) if $\alpha_i = 1$, then $i = k$ and $B_k = C_k$; otherwise B_i is independent and $B_i \cap C_i = \emptyset$.

3 Computation of Bottleneck Decomposition

In this section, we propose a polynomial time algorithm to compute the bottleneck decomposition of any given network. Without loss of generality, we assume that all weights of vertices are positive integers and are bounded by $U > 0$. From Definition 2, it is not hard to see the key to the bottleneck decomposition is the

computation of the maximal bottleneck of each subgraph. But how to find the maximal bottleneck among all of $2^{|V|}$ subsets efficiently is a big challenge. To figure it out, our algorithm comprises two phases on each subgraph. In the first phase, we shall compute the minimal α -ratio α^* of the current subgraph and find the maximal bottleneck with the minimal α -ratio in the second phase. In the subsequent two subsections, we shall introduce the algorithms in each phase detailedly and analyze them respectively.

3.1 Evaluating the Minimal α -ratio α^*

In this phase to evaluate the minimal α -ratio, the main idea of our algorithm is to find the α -ratio by binary search approach iteratively. To reach the minimal ratio, we construct a corresponding network with a parameter α and adjust α by applying the maximum flow algorithm until a certain condition is satisfied. Before proceeding the algorithm to compute the minimal α -ratio, some definitions and lemmas are necessary.

Given a graph $G = (V, E; w)$ and a parameter α , a network $N(G, \alpha)$ (as shown in Fig. 2-(b)) based on G and α is constructed as:

- $V_N = \{s, t\} \cup V \cup \tilde{V}$, where s is the source, t is the sink and \tilde{V} is the copy of V ;
- the directed edge set E_N comprises:
 - a directed edge $(s, v) \in E_N$ from source s to v with capacity of αw_v , $\forall v \in V$;
 - a directed edge $(\tilde{v}, t) \in E_N$ from \tilde{v} to sink t with capacity of w_v , $\forall \tilde{v} \in \tilde{V}$;
 - two directed edges $(u, \tilde{v}) \in E_N$ and $(v, \tilde{u}) \in E_N$ with capacity of ∞ , $\forall (u, v) \in E$.

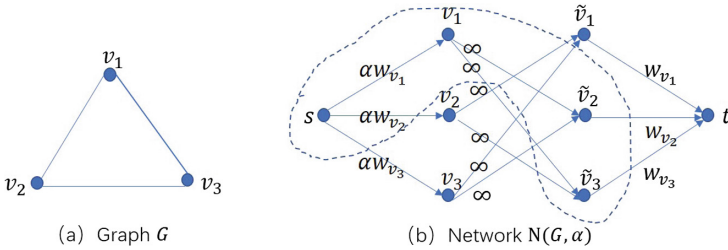


Fig. 2. The illustration of network $N(G, \alpha)$.

In $N(G, \alpha)$, let $\tilde{B} = \{\tilde{v} | v \in B\}$ and $\Gamma(\tilde{B}) = \{\tilde{u} | u \in \Gamma(B)\}$, where the latter is called the neighborhood of \tilde{B} .

Lemma 1. *For any s - t cut (S, T) in network $N(G, \alpha)$, the capacity of cut (S, T) is finite if and only if S has the form as $\{s\} \cup B \cup \Gamma(\tilde{B})$ (as shown in Fig. 2-(b)) for any subset $B \subseteq V$ and its capacity is $\alpha(w(V) - w(B)) + w(\Gamma(B))$.*

Proof. First, if $S = \{s\} \cup B \cup \Gamma(\tilde{B})$, then there are two kinds of directed edges in cut (S, T) :

- edge (s, u) for any $u \notin B$ and the total capacity of these edges is $\alpha(w(V) - w(B))$;
 - edge (\tilde{u}, t) for any $\tilde{u} \in \Gamma(\tilde{B})$ and the total capacity of these edges is $w(\Gamma(B))$.
- So the capacity of cut (S, T) with $S = \{s\} \cup B \cup \Gamma(\tilde{B})$ is finite and its capacity is $\alpha(w(V) - w(B)) + w(\Gamma(B))$. Conversely, if $S \neq \{s\} \cup B \cup \Gamma(\tilde{B})$, then cut (S, T) must contain at least one edge with infinite capacity in the form as (v, \tilde{u}) . At this time the capacity of cut (S, T) is infinite. It completes this claim. \square

Lemma 1 tells that if a cut (S, T) in $N(G, \alpha)$ has a finite capacity, set S must have the form as $S = \{s\} \cup B \cup \Gamma(\tilde{B})$. It is not hard to see that such a finite cut corresponds to a subset $B \subseteq V$. Thus we name B as the *corresponding set* of cut (S, T) . Let $cap(S, T)$ be the capacity of cut (S, T) and $cap(G, \alpha)$ denote the minimum capacity of network $N(G, \alpha)$, that is $cap(G, \alpha) = \min_{(S, T)} cap(S, T)$. To compute the minimal α -ratio, we are more interested in the relationship between the current parameter α and the minimal ratio α^* .

Lemma 2. *Given a graph G and a parameter α . Let α^* be the minimal α -ratio in G and $cap(G, \alpha)$ be the minimum cut capacity of network $N(G, \alpha)$. Then*

- (1) $cap(G, \alpha) < \alpha w(V)$, if and only if $\alpha > \alpha^*$;
- (2) $cap(G, \alpha) = \alpha w(V)$, if and only if $\alpha = \alpha^*$;
- (3) $cap(G, \alpha) > \alpha w(V)$, if and only if $\alpha < \alpha^*$.

Due to the space limit, we leave the proof in full paper. From Lemma 2-(2), the parameter α is equal to α^* , if and only if the corresponding set B of the minimum cut (S, T) in $N(G, \alpha)$ has its α -ratio equal to α , that is $\frac{w(\Gamma(B))}{w(B)} = \alpha = \alpha^*$. Thus we have the following corollary.

Corollary 1. *For the network $N(G, \alpha)$, if the minimum cut (S, T) 's corresponding set B has its α -ratio equal to α , i.e., $\frac{w(\Gamma(B))}{w(B)} = \alpha$, then $\alpha = \alpha^*$.*

Based on Lemmas 1 and 2 and Corollary 1, following Algorithm A is derived to compute the minimal α -ratio for any given graph $G = (V, E; w)$ (Table 1).

The main idea of Algorithm A is to find α^* by binary search approach. The initial range of α^* is set as $[0, 1]$, since $0 < \alpha^* \leq 1$ by Proposition 1-(1). And in each iteration, we split the range in half by comparing the minimum capacity $cap(G, \alpha)$ and the value of $\alpha w(V)$. Because all weights of vertices are positive integers, each subset's α -ratio is a rational number and the difference of any two different α -ratios should be greater than $\frac{1}{w^2(V)}$. So once $\left| \frac{w(\Gamma(B))}{w(B)} - \alpha \right| < \frac{1}{w^2(V)}$,

Table 1. Algorithm A: Compute the minimal α -ratio of G

Input: Graph $G = (V, E; w)$

Output: The minimal α -ratio α^* of G .

1: **Set** $a := 0, b := 1$ and $M := w^2(V)$;

2: **Set** $\alpha := \frac{1}{2}(a + b)$;

3: **Construct** network $N(G, \alpha)$;

4: **Compute** minimum s - t cut (S, T) with its capacity $cap(G, \alpha)$ by Edmonds-Karp algorithm and obtain the corresponding subset $B \subseteq V$;

5: **If** $|\frac{w(\Gamma(B))}{w(B)} - \alpha| < \frac{1}{M}$
 Output $\alpha^* = \frac{w(\Gamma(B))}{w(B)}$;

6: **Else**

7: **If** $cap(G, \alpha) > \alpha w(V)$
 Set $a := \alpha, b := b$ and turn to line 2;

8: **If** $cap(G, \alpha) < \alpha w(V)$
 Set $a := a, b := \alpha$ and turn to line 2.

we can conclude $\frac{w(\Gamma(B))}{w(B)} = \alpha$ and Corollary 1 makes us get $\frac{w(\Gamma(B))}{w(B)} = \alpha = \alpha^*$.

Thus we set the terminal condition of of Algorithm A as $|\frac{w(\Gamma(B))}{w(B)} - \alpha| < \frac{1}{w^2(V)}$.

There is only one loop in Algorithm A. After each round, the length of the search range $[a, b]$ is cut in half. Thus the loop ends within $O(\log(M))$ rounds, that is $O(\log(w(V)))$. In each round, the critical calculation is on line 4 to compute the minimum s - t cut of a given network. Applying the famous *max-flow min-cut theorem* [15], it is equivalent to compute the corresponding maximum flow. There are several polynomial-time algorithms to find maximum flow, for instance *Edmonds-Karp algorithm* [6] with time complexity $O(|V||E|^2)$. In Algorithm A, we call the Edmonds-Karp algorithm to compute the minimum capacity and have the following theorem.

Theorem 1. *The minimal α -ratio α^* of G can be computed in $O(|V||E|^2 \log(w(V)))$ time.*

3.2 Finding the Maximal Bottleneck

In the previous phase, we compute the minimal α -ratio α^* by Algorithm A and the corresponding bottleneck B' also can be obtained. But it is possible that the bottleneck B' may not be maximal. So in this phase, we continue to find the maximal bottleneck given the minimal α -ratio of α^* .

Here we introduce another network, denoted by $N(G, \alpha^*, \epsilon)$, for a given parameter $\epsilon > 0$. To obtain network $N(G, \alpha^*, \epsilon)$, we first construct network $N(G, \alpha^*)$, defined in Sect. 3.1, and then increase the capacity of edge (s, v) , for any $v \in V$, from $\alpha^* w_v$ to $\alpha^* w_v + \epsilon$. By similar proof for Lemma 1, we know a cut (S, T) in $N(G, \alpha^*, \epsilon)$ has a finite capacity, if and only if S has the form as

$S = \{s\} \cup B \cup \Gamma(\tilde{B})$, where $B \subseteq V$. And the corresponding capacity becomes

$$\begin{aligned} \text{cap}(S, T) &= \alpha^*(w(V) - w(B)) + (n - |B|) \cdot \epsilon + w(\Gamma(B)) \\ &= \alpha^*w(V) - w(B)(\alpha^* - \frac{w(\Gamma(B))}{w(B)}) + (n - |B|) \cdot \epsilon. \end{aligned} \quad (1)$$

From (1), the capacity $\text{cap}(S, T)$ actually depends on its corresponding set B and the parameter ϵ . Thus we can view it as a function of B and ϵ . To simplify our discussion in this section, we use $\text{cap}(B, \epsilon)$, different from the notation in the previous subsection, to represent the capacity function of cut (S, T) where $S = \{s\} \cup B \cup \Gamma(\tilde{B})$.

Lemma 3. *Given a graph G , let B^* be the maximal bottleneck of G . For any $\epsilon > 0$, the maximal bottleneck B^* satisfies $\text{cap}(B^*, \epsilon) = \min_{B: \alpha(B) = \alpha^*} \text{cap}(B, \epsilon)$ in network $N(G, \alpha^*, \epsilon)$.*

Proof. For any bottleneck B with $\alpha(B) = \alpha^*$, we know $\text{cap}(B, \epsilon) = \alpha^*w(V) + n \cdot \epsilon - \epsilon|B|$. Thus $\text{cap}(B^*, \epsilon) = \min_{\alpha(B) = \alpha^*} \text{cap}(B, \epsilon)$ since B^* is the maximal bottleneck of G . \square

In other word, if the minimum cut \hat{B} in $N(G, \alpha^*, \epsilon)$, with proper parameter ϵ , is a bottleneck, then $\text{cap}(\hat{B}, \epsilon) = \min_{B \subseteq V} \text{cap}(B, \epsilon) = \min_{B: \alpha(B) = \alpha^*} \text{cap}(B, \epsilon)$, which means \hat{B} is the maximal bottleneck of G .

Corollary 2. *Given a graph G . If there is an $\epsilon > 0$ such that the corresponding set \hat{B} of the minimum cut in $N(G, \alpha^*, \epsilon)$ is a bottleneck, then \hat{B} is the maximal bottleneck of G .*

Furthermore, once the ϵ is small enough, the corresponding set of the minimum cut is the maximal bottleneck of G , shown in Lemma 4. (We leave the proof in full paper due to space limit.)

Lemma 4. *Given a graph G . If $\epsilon \leq \frac{1}{w^3(V)}$, then the corresponding set \hat{B} of the minimum cut in $N(G, \alpha^*, \epsilon)$ is the maximal bottleneck of G .*

Based on Lemma 4, we propose the following Algorithm B to find the maximal bottleneck of a graph G if its minimal α -ration is given beforehand (Table 2).

Lemma 4 guarantees Algorithm B outputs the maximal bottleneck B^* of G correctly. The main body of Algorithm B is to compute the minimum cut capacity of network $N(G, \alpha^*, \epsilon)$ which can be realized by *Edmonds-Karp algorithm*. So the time complexity of Algorithm B is $O(|V||E|^2)$.

Theorem 2. *Algorithm B outputs the maximal bottleneck of G in $O(|V||E|^2)$ time.*

Applying Algorithm A and B, the maximal bottleneck of any given graph can be computed. Thus by Definition 2, we can get the bottleneck decomposition of G by iteratively calling Algorithm A and B on each subgraph. The main result of this paper is:

Table 2. Algorithm B: Find the maximal bottleneck of G

Input: Graph G and its minimal α -ratio α^* ;
Output: The maximal bottleneck B^* of G .

- 1: **Set** $\epsilon := \frac{1}{w^3(V)}$;
- 2: **Construct** network $N(G, \alpha^*, \epsilon)$;
- 3: **Compute** the minimum cut capacity $\text{cap}(\hat{B}, \epsilon)$ by Edmonds-Karp algorithm and obtain the corresponding set $\hat{B} \subseteq V$;
- 4: **Output** $B^* = \hat{B}$;

Theorem 3. *Given a graph $G = (V, E; w)$, the bottleneck decomposition of can be computed in $O(n^6 \log(nU))$ time, where $n = |V|$ and $U = \max_{v \in V} w_v$.*

Proof. To compute the bottleneck decomposition, Algorithm A and B are run repeatedly. In each round we obtain the maximal bottleneck and its neighborhood, then delete them and go to the next round. So the time complexity of each round is $O(|V||E|^2 \log(w(V)))$. At the end of each round at least one vertex is removed. Thus the bottleneck decomposition contains at most $O(|V|)$ loops, which means the total time complexity is $O(|V|^2 |E|^2 \log(w(V)))$. Since $|E|$ is at most $O(|V|^2)$ and the weight of each vertex is bounded by U , the time complexity can be written as $O(|V|^6 \log(|V|U)) = O(n^6 \log(nU))$, if $|V| = n$. \square

4 Bottleneck Decomposition, Market Equilibrium and Fair Allocation

To derive an allocation efficiently, Wu and Zhang [17] modeled the resource exchange system as a pure exchange economy, and obtain the equilibrium allocation by computing a market equilibrium. In this section, we shall present some properties of it, and further prove the allocation of such a market equilibrium not only has the property of proportional response, but also is lex-optimal.

Definition 3 (Market Equilibrium). *Let p_i be the price of agent i 's whole resource, $1 \leq i \leq n$. The price vector $p = (p_1, p_2, \dots, p_n)$, with the allocation $X = (x_{ij})$ is called a market equilibrium if for each agent $i \in V$ the following holds:*

1. $\sum_{j \in \Gamma(i)} x_{ij} = 1$ (market clearance);
2. $\sum_{j \in \Gamma(i)} x_{ji} p_j \leq p_i$ (budget constraint);
3. $X = (x_{ij})$ maximizes $\sum_{j \in \Gamma(i)} x_{ji} w_j$, s.t. $\sum_{j \in \Gamma(i)} x_{ji} p_j \leq p_i$ and $x_{ij} \geq 0$ for each vertex i (individual optimality).

Construction of a Market Equilibrium from Bottleneck Decomposition

Given the bottleneck decomposition $B = \{(B_1, C_1), \dots, (B_k, C_k)\}$, an allocation can be computed by distinguishing three cases [17]. For convenience, such

an allocation mechanism is named as *BD Mechanism* by Cheng *et al.* [2,3]. Figure 3 well illustrates it.

BD Mechanism:

- For $\alpha_i < 1$ (i.e., $B_i \cap C_i = \emptyset$), consider the bipartite graph $\hat{G}_i = (B_i, C_i; E_i)$ where $E_i = (B_i \times C_i) \cap E$. Construct a network by adding source s , sink t and directed edge (s, u) with capacity w_u for any $u \in B_i$, directed edge (v, t) with capacity $\frac{w_v}{\alpha_i}$ for any $v \in C_i$ and directed edge (u, v) with capacity ∞ for any $(u, v) \in E_i$. By the max-flow min-cut theorem, there exists flow $f_{uv} \geq 0$ for $u \in B_i$ and $v \in C_i$ such that $\sum_{v \in \Gamma(u) \cap C_i} f_{uv} = w_u$ and $\sum_{u \in \Gamma(v) \cap B_i} f_{uv} = \frac{w_v}{\alpha_i}$. Let the allocation be $x_{uv} = \frac{f_{uv}}{w_u}$ and $x_{vu} = \frac{\alpha_i f_{vu}}{w_v}$ which means that $\sum_{v \in \Gamma(u) \cap C_i} x_{uv} = 1$ and $\sum_{u \in \Gamma(v) \cap B_i} x_{vu} = \sum_{u \in \Gamma(v) \cap B_i} \frac{\alpha_i \cdot f_{vu}}{w_v} = 1$.
- For $\alpha_k = 1$ (i.e., $B_k = C_k = V_k$), construct a bipartite graph $\hat{G} = (B_k, B'_k; E'_k)$ such that B'_k is a copy of B_k , there is an edge $(u, v') \in E'_k$ if and only if $(u, v) \in E[B_k]$. Construct a network by the above method and by Hall's theorem, for any edge $(u, v') \in E'_k$, there exists flow $f_{uv'}$ such that $\sum_{v' \in \Gamma(u) \cap B'_k} f_{uv'} = w_u$. Let the allocation be $x_{uv} = \frac{f_{uv'}}{w_u}$.
- For any other edge, $(u, v) \notin B_i \times C_i$, $i = 1, 2, \dots, k$, define $x_{uv} = 0$.

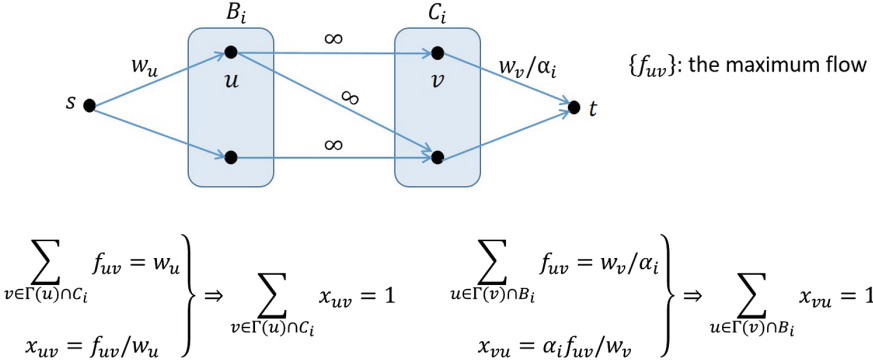


Fig. 3. The illustration of BD Mechanism.

Wu and Zhang stated Proposition 2, saying that once the prices of resource are set properly, such a price vector and the allocation from BD Mechanism make up a market equilibrium, for which the proof is omitted in [17]. To make readers understand clearly, we also propose the detailed proof in full paper.

Proposition 2 ([17]). *Given $B = \{(B_1, C_1), \dots, (B_k, C_k)\}$. If the price to each vertex is set as: for $u \in B_i$, let $p_u = \alpha_i w_u$; and for $u \in C_i$, let $p_u = w_u$, then (p, X) is a market equilibrium, where X is the allocation from BD Mechanism. Furthermore, each agent u 's utility is $U_u = w_u \cdot \alpha_i$, if $u \in B_i$; otherwise $U_u = w_u / \alpha_i$.*

Motivated by P2P systems, such as BitTorrent, the concept of proportional response for the consideration of fairness among all participating agents is put forward to encourage the agents to join in the P2P system.

Definition 4 (Proportional Response). *For each agent i , the allocation $(x_{uv} : v \in \Gamma(u))$ of his resource w_u is proportional to what he receives from his neighbors $(w_v \cdot x_{vu} : v \in \Gamma(u))$, i.e., $x_{uv} = \frac{x_{vu}w_v}{\sum_{k \in \Gamma(u)} x_{ku}w_k} = \frac{x_{vu}w_v}{U_u}$.*

Proposition 3. *The allocation X from BD Mechanism satisfies the property of proportional response.*

Proof. For the allocation from BD Mechanism, if $u \in B_i$ and $v \in C_i$, then $U_u = w_u \cdot \alpha_i$, $U_v = w_v / \alpha_i$ and $f_{uv} = x_{uv} \cdot w_u = \frac{x_{vu} \cdot w_v}{\alpha_i}$. So $x_{uv} = \frac{f_{uv}}{w_u} = \frac{x_{vu}w_v}{\alpha_i w_u} = \frac{x_{vu}w_v}{U_u}$ and $x_{vu} = \frac{\alpha_i f_{vu}}{w_v} = \frac{x_{uv}w_u}{w_v / \alpha_i} = \frac{x_{uv}w_u}{U_v}$. \square

Clearly, the allocation X from BD Mechanism can be computed from the maximum flow in each bottleneck pair (B_i, C_i) , by Edmonds-Karp Algorithm. So the total time complexity of BD Mechanism is $O(n^5)$. Combining Theorem 3 and Proposition 3, we have

Theorem 4. *In the resource sharing system, an allocation with the property of proportional response can be computed in $O(n^6 \log(nU))$.*

Recently, Georgiadis *et al.* [9] discuss the fairness from the compensatory point of view. They characterized the exchange performance of an allocation by the concept of *exchange ratio* vector, in which the coordinate is the exchange ratio $\beta_u(X) = U_u(X)/w_u$ of each agent. In [9], an allocation is said to be fair, if its exchange ratio vector is lex-optimal, and its properties are introduced in the following. Here some notations shall be introduced in advance. Given an allocation X , for a set $S \subseteq V$, $N(S) = \{v \in V : x_{uv} > 0, \exists u \in S\}$ denotes the set of agents who receive resource from agents in S . For the exchange ratio vector $\beta(X)$, the different values (level) of coordinates are denoted by $l_1 < l_2 < \dots < l_M$, $M \leq n$. Let $L_i(X) = \{v \in V : \beta_v(X) = l_i\}$ be the set in which each agent's exchange ratio is equal to l_i . Georgiadis *et al.* [9] proposed the following characterization of a lex-optimal allocation.

Proposition 4 ([9]). *1. An allocation X with $M \geq 2$ is lex-optimal, if and only if*

- (1) L_i is an independent set in G , $i = 1, 2, \dots, \lfloor \frac{M}{2} \rfloor$;
- (2) $L_{M-i+1} = N(L_i)$, $i = 1, 2, \dots, \lfloor \frac{M}{2} \rfloor$;
- (3) $l_i \cdot l_{M-i+1} = 1$, $i = 1, 2, \dots, \lfloor \frac{M}{2} \rfloor$;
- (4) $\sum_{u \in L_i} U_u = \sum_{u \in L_{M-i+1}} w_u$, $i = 1, 2, \dots, \lfloor \frac{M}{2} \rfloor$.

2. An allocation X with $M = 1$ is lex-optimal if and only if $l_1 = 1$.

Based on above proposition, we can continue to conclude that the allocation X from BD Mechanism is also lex-optimal.

Theorem 5. *The allocation X from BD Mechanism is lex-optimal.*

Proof. Given a bottleneck decomposition $\mathcal{B} = \{(B_1, C_1), \dots, (B_k, C_k)\}$. By Proposition 2, $U_u = w_u \cdot \alpha_i$ if $u \in B_i$ and $U_u = w_u/\alpha_i$ if $u \in C_i$. Thus each agent's exchange ratio can be written as: $\beta_u = \alpha_i$ if $u \in B_i$ and $U_u = 1/\alpha_i$ if $u \in C_i$. If $k = 1$ and $B_1 = C_1 = V$, then all agents have the same exchange ratio $\beta_u(X) = 1 = l_1$ with $M = 1$ and the second claim in Proposition 4 is satisfied for this case. If $k = 1$ and $\alpha_k < 1$ or $k > 1$, then we know $\alpha_1 < \alpha_2 < \dots < \alpha_k \leq 1$ by Proposition 1. The relationship between α -ratio and exchange ratio makes the different values of β_u be ordered as: $\alpha_1 < \dots < \alpha_k \leq 1/\alpha_k < \dots < 1/\alpha_1$, where $\alpha_k = 1/\alpha_k$ if and only if $\alpha_k = 1$ and $B_k = C_k$. So the number of different values $M = 2k$ if $\alpha_k < 1$ and $M = 2k - 1$ if $\alpha_k = 1$. By the definitions of l_i and L_i , we have $L_i = B_i$ with $l_i = \alpha_i < 1$ and $L_{M-i+1} = C_i$ with $l_{M-i+1} = 1/\alpha_i > 1$ and $L_i = B_i$ is independent by Proposition 1, $i = 1, \dots, \lfloor M/2 \rfloor$. In addition, since all resource exchange only happens between B_i and C_i by BD Mechanism, $L_{M-i+1} = N(L_i)$ and $\sum_{u \in L_i} U_u = \sum_{v \in L_{M-i+1}} w_v$, $i = 1, \dots, \lfloor M/2 \rfloor$. Until now all statements of the first claim are satisfied for this case. It means the allocation X from BD Mechanism is lex-optimal. \square

5 Conclusion

This paper discusses the issue of the computation of a fair allocation in the resource sharing system through a combinatorial bottleneck decomposition. We design an algorithm to solve the bottleneck decomposition for any graph $G(V, E; w_v)$ in $O(n^6 \log(nU))$ time, where $n = |V|$ and $U = \max_{v \in V} w_v$. Our work also completes the computation of a market equilibrium in the resource exchange system for the consideration of economic efficiency in [17]. Furthermore, we show the equilibrium allocation from the bottleneck decomposition not only is proportional response, but also is lex-optimal, which establishes a connection between two concepts of fairness in [9] and [17]. Involving two different definitions of fairness for resource allocation, we hope to explore other proper concepts of fairness and to design efficient algorithms to find such fair allocations in the future.

References

1. Arrow, K.J., Debreu, G.: Existence of an equilibrium for a competitive economy. *Econometrica: J. Econometric Soc.* 265–290 (1954)
2. Cheng, Y., Deng, X., Pi, Y., Yan, X.: Can bandwidth sharing be truthful? In: Hoefler, M. (ed.) *SAGT 2015. LNCS*, vol. 9347, pp. 190–202. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48433-3_15
3. Cheng, Y., Deng, X., Qi, Q., Yan, X.: Truthfulness of a proportional sharing mechanism in resource exchange. In: *IJCAI*, pp. 187–193 (2016)
4. Duan, R., Garg, J., Mehlhorn, K.: An improved combinatorial polynomial algorithm for the linear Arrow-Debreu market. In: *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 90–106. SIAM (2016)

5. Eaves, B.C.: A finite algorithm for the linear exchange model. Technical report, Systems Optimization Laboratory, Stanford University, California (1975)
6. Edmonds, J., Karp, R.M.: Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM (JACM)* **19**(2), 248–264 (1972)
7. Felson, M., Spaeth, J.L.: Community structure and collaborative consumption: a routine activity approach. *Am. Behav. Sci.* **21**(4), 614–624 (1978)
8. Garg, J., Mehta, R., Sohoni, M., Vazirani, V.V.: A complementary pivot algorithm for market equilibrium under separable, piecewise-linear concave utilities. *SIAM J. Comput.* **44**(6), 1820–1847 (2015)
9. Georgiadis, L., Iosifidis, G., Tassiulas, L.: Exchange of services in networks: competition, cooperation, and fairness. In: *ACM SIGMETRICS Performance Evaluation Review*, vol. 43, pp. 43–56. ACM (2015)
10. Getridapp. <http://getridapp.com>
11. Homeexchange. <http://www.homeexchange.com>
12. Nestia. <http://www.nestia.com>
13. Opengarden. <http://www.opengarden.com>
14. Swap. <http://www.swap.com>
15. Papadimitriou, C.H., Steiglitz, K.: *Combinatorial Optimization: Algorithms and Complexity*. Courier Corporation (1998)
16. Schollmeier, R.: A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. In: *Proceedings of First International Conference on Peer-to-Peer Computing*, pp. 101–102. IEEE (2001)
17. Wu, F., Zhang, L.: Proportional response dynamics leads to market equilibrium. In: *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*, pp. 354–363. ACM (2007)
18. Ye, Y.: A path to the Arrow-Debreu competitive market equilibrium. *Math. Program.* **111**(1–2), 315–348 (2008)



A Local Search $4/3$ -approximation Algorithm for the Minimum 3-path Partition Problem

Yong Chen¹, Randy Goebel², Guohui Lin^{2(✉)}, Longcheng Liu^{2,3}, Bing Su⁴,
Weitian Tong⁵, Yao Xu^{2(✉)}, and An Zhang¹

¹ Department of Mathematics, Hangzhou Dianzi University, Hangzhou, China
{chenyong, anzhang}@hdu.edu.cn

² Department of Computing Science, University of Alberta, Edmonton, Canada
{rgoebel, guohui, xu2}@ualberta.ca

³ School of Mathematical Sciences, Xiamen University, Xiamen, China
longchengliu@xmu.edu.cn

⁴ School of Economics and Management, Xi'an Technological University,
Xi'an, China
subing684@sohu.com

⁵ Department of Computer Science, Georgia Southern University, Statesboro, USA
wtong@georgiasouthern.edu

Abstract. Given a graph $G = (V, E)$, the 3-path partition problem is to find a minimum collection of vertex-disjoint paths each of order at most 3 to cover all the vertices of V . It is different from but closely related to the well-known 3-set cover problem. The best known approximation algorithm for the 3-path partition problem was proposed recently and has a ratio $13/9$. Here we present a local search algorithm and show, by an amortized analysis, that it is a $4/3$ -approximation. This ratio matches up to the best approximation ratio for the 3-set cover problem.

Keywords: k -path partition · Path cover · k -set cover · Approximation algorithms · Local search · Amortized analysis

1 Introduction

Motivated by the data integrity of communication in wireless sensor networks and several other applications, the k -PATH PARTITION (k PP) problem was first considered by Yan et al. [14]. Given a simple graph $G = (V, E)$ (we consider only simple graphs), with $n = |V|$ and $m = |E|$, the *order* of a simple path in G is the number of vertices on the path and it is called a k -path if its order is k . The k PP problem is to find a minimum collection of vertex-disjoint paths each of order at most k such that every vertex is on some path in the collection.

Clearly, the 2PP problem is exactly the MAXIMUM MATCHING problem, which is solvable in $O(m\sqrt{n} \log(n^2/m) / \log n)$ -time [7]. For each $k \geq 3$, k PP is NP-hard [6]. We point out the key phrase “at most k ” in the definition, that

ensures the existence of a feasible solution for any given graph; on the other hand, if one asks for a path partition in which every path has an order exactly k , the problem is called P_k -partitioning and is also NP-complete for any fixed constant $k \geq 3$ [6], even on bipartite graphs of maximum degree three [11]. To the best of our knowledge, there is no approximation algorithm with proven performance for the general k PP problem, except the trivial k -approximation using all 1-paths. For 3PP, Monnot and Toulouse [11] proposed a 3/2-approximation, based on two maximum matchings; recently, Chen et al. [2] presented an improved 13/9-approximation.

The k PP problem is a generalization to the PATH COVER problem [5] (also called PATH PARTITION), which is to find a minimum collection of vertex-disjoint paths which together cover all the vertices in G . PATH COVER contains the HAMILTONIAN PATH problem [6] as a special case, and thus it is NP-hard and it is outside APX unless $P = NP$.

The k PP problem is also closely related to the well-known SET COVER problem. Given a collection of subsets $\mathcal{C} = \{S_1, S_2, \dots, S_m\}$ of a finite ground set $U = \{x_1, x_2, \dots, x_n\}$, an element $x_i \in S_j$ is said to be *covered* by the subset S_j , and a *set cover* is a collection of subsets which together cover all the elements of the ground set U . The SET COVER problem asks to find a minimum set cover. SET COVER is one of the first problems proven to be NP-hard [6], and is also one of the most studied optimization problems for the approximability [8] and inapproximability [4, 12, 13]. The variant of SET COVER in which every given subset has size at most k is called k -SET COVER, which is APX-complete and admits a 4/3-approximation for $k = 3$ [3] and an $(H_k - \frac{196}{390})$ -approximation for $k \geq 4$ [10].

To see the connection between k PP and k -SET COVER, we may take the vertex set V of the given graph as the ground set, and an ℓ -path with $\ell \leq k$ as a subset; then the k PP problem is the same as asking for a minimum *exact* set cover. That is, the k PP problem is a special case of the *minimum EXACT COVER* problem [9], for which unfortunately there is no approximation result that we may borrow. Existing approximations for (non-exact) k -SET COVER do not readily apply to k PP, because in a feasible set cover, an element of the ground set could be covered by multiple subsets. There is a way to enforce the *exactness* requirement in the SET COVER problem, by expanding \mathcal{C} to include all the proper subsets of each given subset $S_j \in \mathcal{C}$. But in an instance graph of k PP, not every subset of vertices on a path is traceable, and so such an expanding technique does not apply. In summary, k PP and k -SET COVER share some similarities, but none contains the other as a special case.

In this paper, we study the 3PP problem. The authors of the 13/9-approximation [2] first presented an $O(nm)$ -time algorithm to compute a k -path partition with the least 1-paths, for any $k \geq 3$; then they applied an $O(n^3)$ -time greedy approach to merge three 2-paths into two 3-paths whenever possible. We aim to design better approximations for 3PP with provable performance, and we achieve a 4/3-approximation. Our algorithm starts with a 3-path partition with the least 1-paths, then it applies a local search scheme to repeatedly search

for an *expected collection* of 2- and 3-paths and replace it by a strictly smaller *replacement collection* of new 2- and 3-paths.

The rest of the paper is organized as follows. In Sect. 2 we present the local search scheme searching for all the expected collections of 2- and 3-paths. The performance of the algorithm is proved through an amortized analysis in Sect. 3. We conclude the paper in Sect. 4.

2 A Local Search Approximation Algorithm

The 13/9-approximation proposed by Chen et al. [2] applies only one replacement operation which is to merge three 2-paths into two 3-paths. In order to design approximation for 3PP with better performance, we examine four more replacement operations each transfers three 2-paths to two 3-paths with the aid of a few other 2- or 3-paths. Starting with a 3-path partition with the least 1-paths, our approximation algorithm repeatedly finds a certain expected collection of 2- and 3-paths and replaces it by a replacement collection of one less new 2- and 3-paths, in which the net gain is exactly one.

In Sect. 2.1 we present all the replacement operations to perform on the 3-path partition with the least 1-paths. The complete algorithm, denoted as APPROX, is summarized in Sect. 2.2.

2.1 Local Operations and Their Priorities

Throughout the local search, the 3-path partitions are maintained to have the least 1-paths. Our four local operations are designed so not to touch the 1-paths, ensuring that the final 3-path partition still contains the least 1-paths. These operations are associated with different priorities, that is, one operation applies only when all the other operations of higher priorities (labeled by smaller numbers) fail to apply to the current 3-path partition. We remind the reader that the local search algorithm is iterative, and every iteration ends after executing a designed local operation. The algorithm terminates when none of the designed local operations applies.

Definition 1. *With respect to the current 3-path partition \mathcal{Q} , a local OPERATION i_1 - i_2 -By- j_1 - j_2 , where $j_1 = i_1 - 3$ and $j_2 = i_2 + 2$, replaces an expected collection of i_1 2-paths and i_2 3-paths of \mathcal{Q} by a replacement collection of j_1 2-paths and j_2 3-paths on the same subset of $2i_1 + 3i_2$ vertices.*

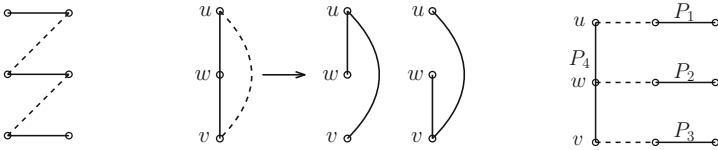
OPERATION 3-0-By-0-2, highest priority 1: When three 2-paths of \mathcal{Q} can be connected into a 6-path in the graph G (see Fig. 1a for an illustration), they form into an expected collection. By removing the middle edge on the 6-path, we achieve two 3-paths on the same six vertices and they form the replacement collection. This is the only local operation executed in the 13/9-approximation [2].

In each of the following operations, we need the aid of one or two 3-paths to transfer three 2-paths to two 3-paths. We first note that for a 3-path u - w - $v \in \mathcal{Q}$,

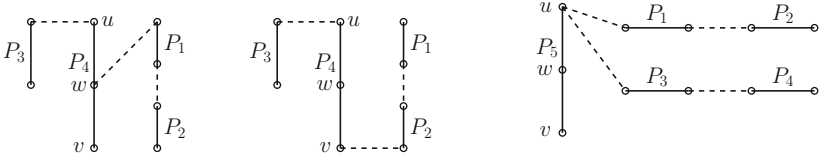
if $(u, v) \in E$ too, then if desired, we may *rotate* $u-w-v$ into another 3-path with w being an endpoint (see Fig. 1b for an illustration). In the following, any 3-path in an expected collection can be either the exact one in \mathcal{Q} or the one rotated from a 3-path in \mathcal{Q} .

OPERATION 3-1-BY-0-3, priority 2: We identify two classes of configurations for the expected collection in this operation. Consider an expected collection of three 2-paths P_1, P_2, P_3 and a 3-path $P_4 = u-w-v$ in \mathcal{Q} .

In the first class, which has priority 2.1, u, w, v are adjacent to an endpoint of P_1, P_2, P_3 in G , respectively (see Fig. 1c for an illustration). The operation breaks the 3-path $u-w-v$ into three singletons and connects each of them to the respective 2-path to form the replacement collection of three new 3-paths.



(a) The configuration of the expected collection for OPERATION 3-0-BY-0-2. (b) A 3-path $u-w-v \in \mathcal{Q}$ can be rotated so that w is an endpoint if $(u, v) \in E$. (c) The first class of configurations of the expected collection for OPERATION 3-1-BY-0-3.



(d) The second class of configurations of the expected collection in OPERATION 3-1-BY-0-3. (e) The configuration of the expected collection for OPERATION 4-1-BY-1-3.

Fig. 1. (a), (c–e) illustrate the configurations of the expected collections for the first three operations, where solid edges are in \mathcal{Q} and dashed edges are in E but outside of \mathcal{Q} . (b) illustrates a rotated 3-path in \mathcal{Q} .

In the second class, which has priority 2.2, two of the three 2-paths, say P_1 and P_2 , are adjacent and thus they can be replaced by a new 3-path and a singleton. We distinguish two configurations in this class (see Fig. 1d for illustrations). In the first configuration, the singleton is adjacent to the midpoint w and P_3 is adjacent to one of u and v ; in the second configuration, the singleton and P_3 are adjacent to u and v , respectively. For an expected collection of either configuration, the operation replaces it by three new 3-paths.

OPERATION 4-1-BY-1-3, priority 3: Consider an expected collection of four 2-paths P_1, P_2, P_3, P_4 and a 3-path $P_5 = u-w-v$ in \mathcal{Q} . These four 2-paths can be separated into two pairs, each of which are adjacent in the graph G , thus we can replace them by two new 3-paths while leaving two singletons. In the

configuration for the expected collection in this operation, the two singletons are adjacent to a common endpoint, say u , of P_5 (see Fig. 1e for an illustration), and they can be replaced by a new 2-path $v-w$ and a new 3-path. Overall, the operation replaces the expected collection by three new 3-paths and a new 2-path.

OPERATION 4-2-BY-1-4, lowest priority 4: Consider an expected collection of four 2-paths P_1, P_2, P_3, P_4 and two 3-paths $P_5 = u-w-v$, $P_6 = u'-w'-v'$ in \mathcal{Q} . These four 2-paths can be separated into two pairs, each of which are adjacent in the graph G , thus we can replace them by two new 3-paths while leaving two singletons. Each of P_5 and P_6 should be adjacent to at least one of the two singletons. We distinguish three classes of configurations for the expected collection in this operation, for which the replacement collection consists of four new 3-paths and a new 2-path.

In the first class, the two singletons are adjacent to P_5 and P_6 at endpoints, say u and u' , respectively; additionally, one of the five edges (u, v') , (v, u') , (w, v') , (v, w') , (v, v') is in E (see Fig. 2a for an illustration).

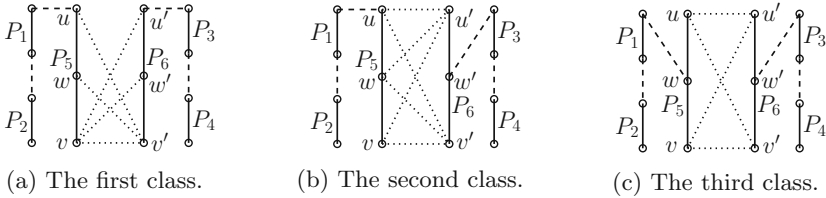


Fig. 2. The three classes of configurations of the expected collections for an OPERATION 4-2-BY-1-4, where solid edges are in \mathcal{Q} , dashed and dotted edges are in E but outside of \mathcal{Q} . In every class, each dotted edge between P_5 and P_6 corresponds to one configuration.

In the second class, one singleton is adjacent to an endpoint of a 3-path, say u on P_5 , and the other singleton is adjacent to the midpoint w' of P_6 ; additionally, one of the six edges (u, u') , (u, v') , (w, u') , (w, v') , (v, u') , (v, v') , is in E (see Fig. 2b for an illustration).

In the third class, the two singletons are adjacent to the midpoints of the two 3-paths, w and w' , respectively; additionally, one of the four edges (u, u') , (u, v') , (v, u') , (v, v') is in E (see Fig. 2c for an illustration).

In each of the above three classes of configurations, the operation replaces P_5, P_6 , and the two singletons by two new 3-paths and one new 2-path.

2.2 The Complete Local Search Algorithm APPROX

A high-level description of the complete algorithm APPROX is depicted in Fig. 3. For the running time, Step 1 takes in $O(nm)$ time [2]. Note that there are $O(n)$ 2-paths and $O(n)$ 3-paths in \mathcal{Q} at the beginning of Step 2, and therefore there

Algorithm APPROX on $G = (V, E)$:

Step 1. compute a 3-path partition \mathcal{Q} with the least 1-paths in G ;

Step 2. Iteratively perform:

- 2.1. if OPERATION 3-0-BY-0-2 applies, update \mathcal{Q} and break;
- 2.2. if OPERATION 3-1-BY-0-3 with priority 2.1 applies, update \mathcal{Q} and break;
- 2.3. if OPERATION 3-1-BY-0-3 with priority 2.2 applies, update \mathcal{Q} and break;
- 2.4. if OPERATION 4-1-BY-1-3 applies, update \mathcal{Q} and break;
- 2.5. if OPERATION 4-2-BY-1-4 applies, update \mathcal{Q} and break;

Step 3. Return \mathcal{Q} .

Fig. 3. A high-level description of the algorithm APPROX, where each “break” ends the current iteration of Step 2.

are $O(n^6)$ original candidate collections to be examined, since a candidate collection has a maximum size of 6. When a local operation applies, an iteration ends and the 3-path partition \mathcal{Q} reduces its size by 1, while introducing at most 5 new 2- and 3-paths. These new 2- and 3-paths give rise to $O(n^5)$ new candidate collections to be examined in the subsequent iterations. Since there are at most n iterations in Step 2, we conclude that the total number of original and new candidate collections examined in Step 2 is $O(n^6)$. Determining whether a candidate collection is an expected collection, and if so, deciding the corresponding replacement collection, can be done in $O(1)$ time. We thus prove that the overall running time of Step 2 is $O(n^6)$, and consequently prove the following theorem.

Theorem 1. *The running time of the algorithm APPROX is in $O(n^6)$.*

3 Analysis of the Approximation Ratio 4/3

In this section, we show that our local search algorithm APPROX is a 4/3-approximation for 3PP. The performance analysis is done through amortization.

The 3-path partition produced by the algorithm APPROX is denoted as \mathcal{Q} ; let \mathcal{Q}_i denote the sub-collection of i -paths in \mathcal{Q} , for $i = 1, 2, 3$, respectively. Let \mathcal{Q}^* be an optimal 3-path partition, *i.e.*, it achieves the minimum total number of paths, and let \mathcal{Q}_i^* denote the sub-collection of i -paths in \mathcal{Q}^* , for $i = 1, 2, 3$, respectively. Since our \mathcal{Q} contains the least 1-paths among all 3-path partitions for G , we have $|\mathcal{Q}_1| \leq |\mathcal{Q}_1^*|$. Since both \mathcal{Q} and \mathcal{Q}^* cover all the vertices of V , we have $|\mathcal{Q}_1| + 2|\mathcal{Q}_2| + 3|\mathcal{Q}_3| = n = |\mathcal{Q}_1^*| + 2|\mathcal{Q}_2^*| + 3|\mathcal{Q}_3^*|$.

Next, we prove the following inequality which gives an upper bound on $|\mathcal{Q}_2|$, through an amortized analysis:

$$|\mathcal{Q}_2| \leq |\mathcal{Q}_1^*| + 2|\mathcal{Q}_2^*| + |\mathcal{Q}_3^*|. \quad (1)$$

It follows that $3|\mathcal{Q}_1| + 3|\mathcal{Q}_2| + 3|\mathcal{Q}_3| \leq 4|\mathcal{Q}_1^*| + 4|\mathcal{Q}_2^*| + 4|\mathcal{Q}_3^*|$, that is, $|\mathcal{Q}| \leq \frac{4}{3}|\mathcal{Q}^*|$, and consequently the following theorem holds.

Theorem 2. *The algorithm APPROX is an $O(n^6)$ -time $4/3$ -approximation for the 3PP problem, and the performance ratio $4/3$ is tight for APPROX.*

In the amortized analysis, each 2-path of \mathcal{Q}_2 has one token (*i.e.*, $|\mathcal{Q}_2|$ tokens in total) to be distributed to the paths of \mathcal{Q}^* . The upper bound in Eq. (1) will immediately follow if we prove the following lemma.

Lemma 1. *There is a distribution scheme in which*

1. every 1-path of \mathcal{Q}_1^* receives at most 1 token;
2. every 2-path of \mathcal{Q}_2^* receives at most 2 tokens;
3. every 3-path of \mathcal{Q}_3^* receives at most 1 token.

In the rest of the section we present the distribution scheme that satisfies the three requirements stated in Lemma 1.

Denote $E(\mathcal{Q}_2)$, $E(\mathcal{Q}_3)$, $E(\mathcal{Q}_2^*)$, $E(\mathcal{Q}_3^*)$ as the set of all the edges on the paths of \mathcal{Q}_2 , \mathcal{Q}_3 , \mathcal{Q}_2^* , \mathcal{Q}_3^* , respectively, and $E(\mathcal{Q}^*) = E(\mathcal{Q}_2^*) \cup E(\mathcal{Q}_3^*)$. In the subgraph of $G(V, E(\mathcal{Q}_2) \cup E(\mathcal{Q}^*))$, only the midpoint of a 3-path of \mathcal{Q}_3^* may have degree 3, *i.e.*, incident with two edges of $E(\mathcal{Q}^*)$ and an edge of $E(\mathcal{Q}_2)$, while all the other vertices have degree at most 2 since each is incident with at most one edge of $E(\mathcal{Q}_2)$ and at most one edge of $E(\mathcal{Q}^*)$.

Our distribution scheme consists of two phases. We define two functions $\tau_1(P)$ and $\tau_2(P)$ to denote the fractional amount of token received by a path $P \in \mathcal{Q}^*$ in Phase 1 and Phase 2, respectively; we also define the function $\tau(P) = \tau_1(P) + \tau_2(P)$ to denote the total amount of token received by the path $P \in \mathcal{Q}^*$ at the end of our distribution process. Then, we have $\sum_{P \in \mathcal{Q}^*} \tau(P) = |\mathcal{Q}_2|$.

3.1 Token Distribution Phase 1

In Phase 1, we distribute all the $|\mathcal{Q}_2|$ tokens to the paths of \mathcal{Q}^* (*i.e.*, $\sum_{P \in \mathcal{Q}^*} \tau_1(P) = |\mathcal{Q}_2|$) such that a path $P \in \mathcal{Q}^*$ receives some token from a 2-path $u-v \in \mathcal{Q}_2$ only if u or v is (or both are) on P , and the following three requirements are satisfied:

1. $\tau_1(P_i) \leq 1$ for $\forall P_i \in \mathcal{Q}_1^*$;
2. $\tau_1(P_j) \leq 2$ for $\forall P_j \in \mathcal{Q}_2^*$;
3. $\tau_1(P_\ell) \leq 3/2$ for $\forall P_\ell \in \mathcal{Q}_3^*$.

In this phase, the one token held by each 2-path of \mathcal{Q}_2 is breakable but can only be broken into two halves. Thus for every path $P \in \mathcal{Q}^*$, $\tau_1(P)$ is a multiple of $1/2$.

For each 2-path $u-v \in \mathcal{Q}_2$, at most one of u and v can be a singleton of \mathcal{Q}^* . If $P_1 = v \in \mathcal{Q}_1^*$, then the whole 1 token of the path $u-v$ is distributed to v , that is, $\tau_1(v) = 1$ (see Fig. 4a for an illustration). This way, we have $\tau_1(P) \leq 1$ for $\forall P \in \mathcal{Q}_1^*$.

For a 2-path $u-v \in \mathcal{Q}_2$, we consider the cases when both u and v are incident with an edge of $E(\mathcal{Q}^*)$. If one of u and v , say v , is incident with an edge of $E(\mathcal{Q}_2^*)$, that is, v is on a 2-path $P_1 = v-w \in \mathcal{Q}_2^*$, then the 1 token of the path $u-v$

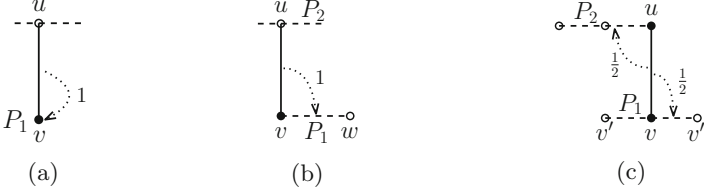


Fig. 4. Illustrations of the token distribution scheme in Phase 1, where solid edges are in $E(Q_2)$ and dashed edges are in $E(Q^*)$. In (c), u or v can be either an endpoint or the midpoint of the corresponding 3-path of Q_3^* .

is given to the path $P_1 \in Q_2^*$ (see Fig. 4b for an illustration). Note that if u is also on a 2-path $P_2 \in Q_2^*$ and $P_2 \neq P_1$, then the path P_2 receives no token from the path $u-v$. The choice of which of the two vertices u and v comes first does not matter. This way, we have $\tau_1(P) \leq 2$ for $\forall P \in Q_2^*$ since the 2-path $P_1 \in Q_2^*$ might receive another token from a 2-path of Q_2 incident at w .

Next, we consider the cases for a 2-path $u-v \in Q_2$ in which each of u and v is incident with an edge of $E(Q_3^*)$. Consider a 3-path $P_1 \in Q_3^*$: $v'-v-v''$. We distinguish two cases for a vertex of P_1 to determine the amount of token received by P_1 (see Fig. 4c for an illustration). In the first case, either the vertex, say v' , is not on any path of Q_2 or it is on a path of Q_2 with 0 token left, then P_1 receives no token *through vertex* v' . In the second case, the vertex, say v (the following argument also applies to the other two vertices v' and v''), is on a path $u-v \in Q_2$ holding 1 token, and consequently u must be on a 3-path $P_2 \in Q_3^*$, then the 1 token of $u-v$ is broken into two halves, with $1/2$ token distributed to P_1 through vertex v and the other $1/2$ token distributed to P_2 through vertex u . This way, we have $\tau_1(P) \leq 3/2$ for $\forall P \in Q_3^*$ since the 3-path $P_1 \in Q_3^*$ might receive another $1/2$ token through each of v' and v'' .

3.2 Token Distribution Phase 2

In Phase 2, we will transfer the extra $1/2$ token from every 3-path $P \in Q_3^*$ with $\tau_1(P) = 3/2$ to some other paths of Q^* in order to satisfy the three requirements of Lemma 1. In this phase, each $1/2$ token can be broken into two quarters, thus for a path $P \in Q^*$, $\tau_2(P)$ is a multiple of $1/4$.

Consider a 3-path $P_1 = v''-v'-v \in Q_3^*$. We observe that if $\tau_1(P_1) = 3/2$, then each of v , v' , and v'' must be incident with an edge of $E(Q_2)$, the other endpoint of which must also be on a 3-path of Q_3^* . One of the three vertices, say v , on an edge $(u, v) \in E(Q_2)$, must have its corresponding u outside of P_1 . Denote P_2 as the 3-path of Q_3^* where u is on. Let w be a vertex adjacent to u on P_2 . We can verify that due to Q being a partition with the least 1-paths and by OPERATION 3-0-BY-0-2, w cannot be a singleton of Q_1 or on any 2-path of Q_2 , and thus it must be on a 3-path of Q_3 , being either an endpoint or the midpoint (see Fig. 5 for an illustration). We thus conclude that $\tau_1(P_2) \leq 1$, and we have the following lemma.

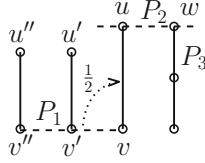


Fig. 5. An illustration of a 3-path $P_1 = v-v'-v'' \in \mathcal{Q}_3^*$ with $\tau_1(P_1) = 3/2$, where $u-v, u'-v', u''-v'' \in E(\mathcal{Q}_2)$, $P_3 \in \mathcal{Q}_3$, with w being either the midpoint or an endpoint of P_3 , and $P_2 \in E(\mathcal{Q}_3^*)$ is represented by dashed edges, on which w is adjacent to u .

Lemma 2. For any 3-path $P_1 \in \mathcal{Q}_3^*$ with $\tau_1(P_1) = 3/2$, there must be another 3-path $P_2 \in \mathcal{Q}_3^*$ with $\tau_1(P_2) \leq 1$ such that

1. $u-v$ is a 2-path of \mathcal{Q}_2 , where v is on P_1 and u is on P_2 , and
2. any vertex adjacent to u on P_2 must be on a 3-path P_3 of \mathcal{Q}_3 .

The first step of Phase 2 is to transfer this extra $1/2$ token back from P_1 to the 2-path $u-v$ through vertex v (see Fig. 5 for an illustration). Thus, we have $\tau_2(P_1) = -1/2$ and $\tau(P_1) = 3/2 - 1/2 = 1$.

Using Lemma 2 and its notation, let x_1 and y_1 be the other two vertices on P_3 ($P_3 = w-x_1-y_1$ or $P_3 = x_1-w-y_1$). Denote $P_4 \in \mathcal{Q}^*$ ($P_5 \in \mathcal{Q}^*$, respectively) as the path where x_1 (y_1 , respectively) is on. Next, we will transfer the $1/2$ token from $u-v$ to the paths P_4 or/and P_5 through some pipe or pipes.

We define a pipe $r \rightarrow s \rightarrow t$, where r is an endpoint of a 2-path of \mathcal{Q}_2 which receives $1/2$ token in the first step of Phase 2, (r, s) is an edge on a 3-path $P' \in \mathcal{Q}_3^*$ with $\tau_1(P') \leq 1$ ($P' = P_2$ here), s and t are both on a 3-path of \mathcal{Q}_3 (P_3 here), and t is a vertex on our destination path of \mathcal{Q}^* (P_4 or P_5 here) which will receive token from the 2-path of \mathcal{Q}_2 . r and t are called the head and tail of the pipe, respectively. For example, in Fig. 6a, there are four possible pipes $u \rightarrow w \rightarrow x_1$, $u \rightarrow w \rightarrow y_1$, $u'' \rightarrow w \rightarrow x_1$, and $u'' \rightarrow w \rightarrow y_1$. We distinguish the cases, in which the two paths P_4 and P_5 belong to different combinations of $\mathcal{Q}_1^*, \mathcal{Q}_2^*, \mathcal{Q}_3^*$, to determine how they receive more token through some pipe or pipes.

Recall that u can be either an endpoint or the midpoint of P_2 . We discuss the cases with u being an endpoint of P_2 (the cases for u being the midpoint can be discussed the same), that is, $P_2 = u-w-u''$. The following is a summary of our discussion and results ([1] contains the full details).

Case 1. At least one of P_4 and P_5 is a singleton of \mathcal{Q}_1^* , say $P_4 = x_1 \in \mathcal{Q}_1^*$ (see Fig. 6 for illustrations). In this case, $\tau_1(P_4) = 0$, and we transfer the $1/2$ token from $u-v$ to P_4 through pipe $u \rightarrow w \rightarrow x_1$. We prove in [1] that there are at most two pipes with tail x_1 through each of which could P_4 receive $1/2$ token, due to OPERATION 3-1-BY-0-3 and OPERATION 4-1-BY-1-3. That is, we have $\tau_2(P_4) \leq 1/2 \times 2 = 1$, implying $\tau(P_4) \leq 0 + 1 = 1$.

Case 2. Both P_4 and P_5 are paths of $\mathcal{Q}_2^* \cup \mathcal{Q}_3^*$. In this case, if w is an endpoint of $P_3 = w-x_1-y_1$, with y_1 on P_5 , we transfer the $1/2$ token from $u-v$ to P_5 through

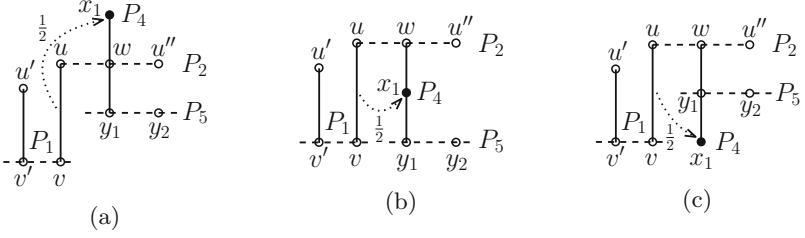


Fig. 6. The cases when P_4 is a singleton of \mathcal{Q}_1^* , where solid edges are in $E(\mathcal{Q}_2)$ or $E(\mathcal{Q}_3)$ and dashed edges are in $E(\mathcal{Q}^*)$. x_1 is the tail of the pipe through which P_4 could receive $1/2$ token from the 2-path of $u-v$.

pipe $u \rightarrow w \rightarrow y_1$ (see Fig. 7a for an illustration); if w is the midpoint of P_3 , we transfer $1/4$ token from $u-v$ to P_4 through pipe $u \rightarrow w \rightarrow x_1$ and the other $1/4$ token to P_5 through pipe $u \rightarrow w \rightarrow y_1$ (see Fig. 7b for an illustration). We prove in [1] that for any $P \in \{P_4, P_5\}$, if $\tau_2(P) > 0$, then we have $\tau_1(P) \leq 1/2$ and $\tau_2(P) \leq 1$, and it falls into one of the following four cases:

1. If w is an endpoint of P_3 and $\tau_1(P) = 0$, then there are at most two pipes through each of which could P receive $1/2$ token. That is, $\tau_2(P) \leq 1/2 \times 2 = 1$, implying $\tau(P) \leq 0 + 1 = 1$.
2. If w is an endpoint of P_3 and $\tau_1(P) = 1/2$, then only through one pipe could P receive the $1/2$ token. That is, $\tau_2(P) \leq 1/2$, implying $\tau(P) \leq 1/2 + 1/2 = 1$.
3. If w is the midpoint of P_3 and $\tau_1(P) = 0$, then there are at most four pipes through each of which could P receive $1/4$ token. That is, $\tau_2(P) \leq 1/4 \times 4 = 1$, implying $\tau(P) \leq 0 + 1 = 1$.
4. If w is the midpoint of P_3 and $\tau_1(P) = 1/2$, then there are at most two pipes through each of which could P receive $1/4$ token. That is, $\tau_2(P) \leq 1/4 \times 2 = 1/2$, implying $\tau(P) \leq 1/2 + 1/2 = 1$.

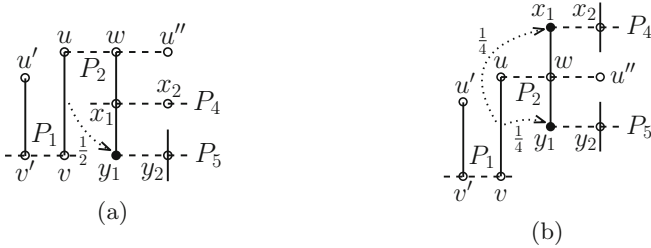


Fig. 7. The cases when both P_4 and P_5 are in $\mathcal{Q}_2^* \cup \mathcal{Q}_3^*$, where solid edges are in $E(\mathcal{Q}_2)$ or $E(\mathcal{Q}_3)$ and dashed edges are in $E(\mathcal{Q}^*)$. In (a), y_1 is the tail of the pipe through which P_5 receives $1/2$ token from the 2-path $u-v$; in (b), x_1 is the tail of the pipe through which P_4 receives $1/4$ token from the 2-path $u-v$ and y_1 is the tail of the pipe through which P_5 receives $1/4$ token from the 2-path $u-v$.

In summary, for any $P_1 \in \mathcal{Q}^*$ with $\tau_1(P_1) = 3/2$, we have $\tau_2(P_1) = -1/2$; for any $P \in \mathcal{Q}^*$ with $\tau_2(P) > 0$, we have $\tau_1(P) = 0$ if $\tau_2(P) \leq 1$, or $\tau_1(P) \leq 1/2$ if $\tau_2(P) \leq 1/2$. Therefore, at the end of Phase 2, we have

1. $\tau(P_i) \leq 1$ for $\forall P_i \in \mathcal{Q}_1^*$,
2. $\tau(P_j) \leq 2$ for $\forall P_j \in \mathcal{Q}_2^*$,
3. $\tau(P_\ell) \leq 1$ for $\forall P_\ell \in \mathcal{Q}_3^*$.

This proves Lemma 1.

3.3 A Tight Instance for APPROX

Figure 8 illustrates a tight instance, in which our solution 3-path partition \mathcal{Q} contains nine 2-paths and three 3-paths (solid edges) and an optimal 3-path partition \mathcal{Q}^* contains nine 3-paths (dashed edges). Each 3-path of \mathcal{Q}^* receives 1 token from the 2-paths in \mathcal{Q} in our distribution process. This instance shows that the performance ratio of $4/3$ is tight for APPROX.

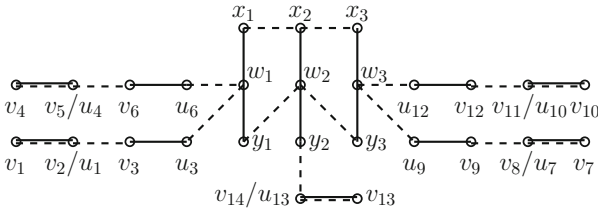


Fig. 8. A tight instance of 27 vertices, where solid edges represent a 3-path partition produced by APPROX and dashed edges represent an optimal 3-path partition. The edges (u_{3i+1}, v_{3i+1}) , $i = 0, 1, \dots, 4$, are in $E(\mathcal{Q}_2) \cap E(\mathcal{Q}^*)$, shown in both solid and dashed. The vertex u_{3i+1} collides into v_{3i+2} , $i = 0, 1, \dots, 4$. In our distribution process, each of the nine 3-paths in \mathcal{Q}^* receives 1 token from the 2-paths in \mathcal{Q} .

Lemma 1 and the tight instance shown above together prove Theorem 2.

4 Conclusions

We studied the 3PP problem and designed a $4/3$ -approximation algorithm APPROX. APPROX first computes a 3-path partition \mathcal{Q} with the least 1-paths in $O(nm)$ -time, then iteratively applies four local operations with different priorities to reduce the total number of paths in \mathcal{Q} . The overall running time of APPROX is $O(n^6)$. The performance ratio $4/3$ of APPROX is proved through an amortization scheme, using the structure properties of the 3-path partition returned by APPROX. We also show that the performance ratio $4/3$ is tight for our algorithm.

The 3PP problem is closely related to the 3-SET COVER problem, but none is a special case of the other. The best 4/3-approximation for 3-SET COVER has stood there for more than three decades; our algorithm APPROX for 3PP has the approximation ratio matches up to this best approximation ratio 4/3. We leave it open to better approximate 3PP.

Acknowledgement. YC and AZ were supported by the NSFC Grants 11771114 and 11571252; YC was also supported by the China Scholarship Council Grant 201508330054. RG, GL and YX were supported by the NSERC Canada. LL was supported by the China Scholarship Council Grant No. 201706315073, and the Fundamental Research Funds for the Central Universities Grant No. 20720160035. WT was supported in part by funds from the College of Engineering and Computing at the Georgia Southern University.

References

1. Chen, Y., et al.: A local search 4/3-approximation algorithm for the minimum 3-path partition problem. [arXiv:1812.09353](https://arxiv.org/abs/1812.09353) (2018)
2. Chen, Y., Goebel, R., Lin, G., Su, B., Xu, Y., Zhang, A.: An improved approximation algorithm for the minimum 3-path partition problem. *J. Comb. Optim.* (2018). <https://doi.org/10.1007/s10878-018-00372-z>
3. Duh, R., Fürer, M.: Approximation of k -set cover by semi-local optimization. In: *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing (STOC 1997)*, pp. 256–264 (1997)
4. Feige, U.: A threshold of for approximating set cover. *J. ACM* **45**, 634–652 (1998)
5. Franzblau, D.S., Raychaudhuri, A.: Optimal hamiltonian completions and path covers for trees, and a reduction to maximum flow. *ANZIAM J.* **44**, 193–204 (2002)
6. Garey, M.R., Johnson, D.S.: *Computers and Intractability: A Guide to the Theory of NP-completeness*. W. H. Freeman and Company, San Francisco (1979)
7. Goldberg, A.V., Karzanov, A.V.: Maximum skew-symmetric flows and matchings. *Math. Program.* **100**, 537–568 (2004)
8. Johnson, D.S.: Approximation algorithms for combinatorial problems. *J. Comput. Syst. Sci.* **9**, 256–278 (1974)
9. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W., Bohlinger, J.D. (eds.) *Complexity of Computer Computations*, pp. 85–103. Springer, Boston (1972). https://doi.org/10.1007/978-1-4684-2001-2_9
10. Levin, A.: Approximating the unweighted k -set cover problem: greedy meets local search. In: Erlebach, T., Kaklamanis, C. (eds.) *WAOA 2006. LNCS*, vol. 4368, pp. 290–301. Springer, Heidelberg (2007). https://doi.org/10.1007/11970125_23
11. Monnot, J., Toulouse, S.: The path partition problem and related problems in bipartite graphs. *Oper. Res. Lett.* **35**, 677–684 (2007)
12. Raz, R., Safra, S.: A sub-constant error-probability low-degree test, and sub-constant error-probability PCP characterization of NP. In: *Proceedings of the 29th Annual ACM Symposium on Theory of Computing (STOC 1997)*, pp. 475–484 (1997)
13. Vazirani, V.: *Approximation Algorithms*. Springer, Heidelberg (2001). <https://doi.org/10.1007/978-3-662-04565-7>
14. Yan, J.-H., Chang, G.J., Hedetniemi, S.M., Hedetniemi, S.T.: k -path partitions in trees. *Discrete Appl. Math.* **78**, 227–233 (1997)



Efficient Guarding of Polygons and Terrains

Pradeesha Ashok^(✉) and Meghana M. Reddy

International Institute of Information Technology Bangalore, Bengaluru, India
pradeesha@iiitb.ac.in, meghanam.reddy@iiitb.org

Abstract. In this paper, we study the EFFICIENT GUARDING problem - a variant of the well studied ART GALLERY Problem in computational geometry. A given polygon P is considered to be guarded efficiently by a guard set G if every point in P is seen by exactly one guard in G . Here we investigate the problem of efficient guarding of all the vertices of a polygon using a vertex guard set of minimum size. We prove that it is NP-complete even to check whether an efficient guard set exists for a polygon. We then give a parameterized algorithm for the efficient guarding of a 1.5 dimensional terrain, when parameterized by a structural parameter namely, the onion peeling number of the terrain i.e, the number of convex layers of the terrain. We further give polynomial time algorithms to solve the minimum efficient guarding problem for some special polygons.

Keywords: Art Gallery Problem · Efficient Guarding · FPT algorithms

1 Introduction

The ART GALLERY problem is among the well studied problems in the field of computational geometry. The first question to be studied in this field was the following: Given a simple polygon P of n vertices, how many guards (points) need to be placed inside P such that every point in P is seen by one of the guards? Here, two points are said to see each other if the line segment connecting them lies entirely inside P . It was proved that any polygon on n vertices can be guarded using $\frac{n}{3}$ guards [9]. It can also be seen that for some polygons $\frac{n}{3}$ guards become necessary.

Apart from the theoretical interest, the ART GALLERY problem is also relevant due to the large number of applications. A number of variants and special cases of ART GALLERY problem are studied. These are broadly defined by the positions where guards can be placed, the properties of the polygon to be guarded and the definition of visibility. See [29] for a detailed survey.

An important aspect of any variant of ART GALLERY problem is the algorithmic question of finding a guard set of minimum size. This was proved to be NP-Hard [24] and later APX-Hard [12] for general polygons. The minimum ART

GALLERY problem is known to be NP-Hard even when the polygons are orthogonal [30], monotone [23] and 1.5 dimensional terrains [22]. Art Gallery Problem is extensively studied in the context of approximation algorithms [17, 18, 21].

The minimum guarding problem is also studied in the area of fixed parameterized tractability and exact algorithms. Minimum guarding problem is shown to be W-hard [7] and therefore no FPT algorithms are expected to exist under standard parameterization i.e., when the parameter is solution size. Parameterized algorithms are particularly well studied in the terrain guarding problem. FPT algorithms for terrain guarding parameterized by the onion peeling number [19] and guard range [20] are known. Also, orthogonal terrain guarding is FPT parameterized by the solution size [2].

Efficient Guarding Problem: In this paper, we study a variant of the ART GALLERY problem which is motivated by applications in wireless networks, robotics etc. In particular, this restriction is meaningful in applications where intersection of ranges of two guards leads to noise.

Definition 1. *Let P be a polygon with n vertices. Let $C \subseteq P$ represent a set of points in P that needs to be guarded and G represent a set of points in P where guards are allowed to be placed. We call a subset $G' \subseteq G$ an efficient guard set for C if every point in C is seen by exactly one guard in G' . Equivalently, C is said to be guarded efficiently by G' .*

A minimum efficient guard set is an efficient guard set of minimum size possible. The minimum EFFICIENT GUARDING problem is to find a minimum efficient guard set. In the case where both C and G is the entire polygon P , then the EFFICIENT GUARDING problem is same as testing whether P is a star polygon i.e., checking whether there exists a point that can see the entire polygon. This can be done in linear time [27]. Similarly if $C = P$ and $G = V$, the problem reduces to checking whether there exists a vertex of P that sees the entire polygon which can be solved in polynomial time [29].

In this paper we consider the problem of efficient guarding of every vertex of the polygon P using guards placed at vertices of P .

EFFICIENT GUARDING: Given a polygon P with vertex set V , $|V| = n$, find $G \subset V$ such that every vertex $v \in V$ is seen by exactly one vertex in G . This problem is not studied before, to the best of our knowledge.

Related Problems and Results: EFFICIENT GUARDING problem is closely related to the CHROMATIC ART GALLERY problem and the CONFLICT FREE ART GALLERY problem. In the CHROMATIC ART GALLERY problem, the polygon needs to be guarded by G and there exists a k -coloring of G such that all the guards that see a point p in the polygon are of different colors, for all p that needs to be guarded. In the CONFLICT FREE ART GALLERY problem, the polygon needs to be guarded by G and there exists a k -coloring of G such that every point in P that needs to be guarded is seen by a guard of distinct color. In both these problems, we are trying to minimize the number of colors needed. In EFFICIENT GUARDING problem, we are trying to minimize the number of

guards when the number of colors used in chromatic and conflict free art gallery problems is fixed as one. Chromatic and conflict-free art gallery problems are studied, with an emphasis on bounding the minimum number of colors required in [4, 5]. The algorithmic question of minimizing the number of colors for a given polygon is also studied [13, 15].

Guarding problems can also be linked to the area of visibility graphs. The problem of guarding all vertices of a polygon using minimum number of vertex guards is equivalent to the minimum dominating set problem in the polygon visibility graphs. Similarly, the EFFICIENT GUARDING problem of vertices using vertex guards is equivalent to the EFFICIENT DOMINATION problem.

Definition 2 [3]. *Given a graph $G(V, E)$, $D \subseteq V$ is an efficient dominating set if for every vertex $v \in V$, $|N[v] \cap D| = 1$.*

The EFFICIENT DOMINATION problem has found applications in areas like coding theory, facility location, graph embedding etc. The problem of finding an efficient dominating set of minimum size is NP-hard. Hardness results and polynomial time algorithms are studied for various graph classes [6, 25, 26].

2 Preliminaries

In this section, we discuss some concepts and results that will be used in the subsequent sections.

Fixed-Parameter Tractability: Under standard complexity theoretical assumptions, NP-hard and NP-complete problems are not expected to have polynomial time algorithms. Here, we try to design algorithms which solve the problem exactly and have exponential running time but the exponential factor in the running time is restricted to a parameter which is assumed to be small. A problem instance π , with a parameter k , is called *Fixed Parameter Tractable* if there exists an algorithm that solves the problem in $f(k) \cdot |\pi|^c$, where c is a constant and $f(k)$ is a computable function independent of π . The parameter k is a small positive integer which can be a structural property of either the input or output of π . The running time of FPT algorithms turns out to be efficient compared to exponential running time algorithms. FPT algorithms and the various techniques can be studied from [10, 11].

Visibility Graphs: Visibility graph of a polygon is constructed to capture the visibility between different vertices of the polygon. The vertex set of the visibility graph corresponds to the vertex set of the polygon and an edge is added between two vertices in the visibility graph if the corresponding vertices in the polygon see each other. Visibility graphs and visibility algorithms are well studied. For details, see [16].

Treewidth: Tree decomposition of a graph G is a pair (T, X) , where T is a tree, and $X_t \subseteq V(G)$ is a vertex subset, where t is a node in the tree T . X_t is called bag of t , and the following three conditions hold:

- Every vertex of the graph G is in at least one bag.
- For every edge $uv \in E(G)$, there is at least one node t of tree T such that both u and v belong to X_t .
- For every vertex $v \in V(G)$, the set of nodes of T whose corresponding bags contain v , induces a connected subtree of T .

The *width* of a tree decomposition is one less than the maximum size of any bag, i.e., $\max_t |X_t| - 1$. The *treewidth* of a graph G is the minimum possible width of a tree decomposition of G , and it is denoted by $\tau(G)$.

Tree decomposition of a graph is very useful in solving problems. A well-known approach is applying dynamic programming over the tree decomposition of the graph while using the three properties to define the recursion. This technique gives an FPT algorithm for problems like dominating set, vertex cover etc.

Nice Tree Decomposition: A tree decomposition with a distinguished root is called a nice tree decomposition if:

- All the leaf nodes and the root node contain empty bags, i.e., $X_l = X_r = \phi$, where r is the root node and l is a leaf node.
- Every other node of the tree decomposition falls in one of the three categories:
- **Introduce node:** An introduce vertex node t has one child t' such that $X_t = X_{t'} \cup \{v\}$ for some $v \notin X_{t'}$.
- **Forget node:** A forget node t has one child t' such that $X_t = X_{t'} \setminus \{w\}$ for some $w \in X_{t'}$.
- **Join node:** A join node t has two children t_1 and t_2 , such that $X_t = X_{t_1} = X_{t_2}$.

3 NP-Hardness

We show that it is NP-complete even to check whether an efficient guard set exists. Membership in NP is obvious. We reduce POSITIVE PLANAR 1-IN-3-SAT to EFFICIENT GUARDING in polygons. POSITIVE PLANAR 1-IN-3-SAT was introduced and shown to be NP-complete by Mulzer and Rote [28].

Definition 3. POSITIVE PLANAR 1-IN-3-SAT: A formula ϕ in 3-CNF is called *positive* if and only if it does not contain any negation, i.e., if and only if all occurring literals are positive. A formula ϕ with clause set $C = \{c_1, \dots, c_l\}$ and variable set $X = \{x_1, \dots, x_n\}$ is called *backbone planar* if and only if its associated graph $G(\phi) = (X \cup C, E(\phi))$ is planar, where $E(\phi)$ is defined as

- $x_i c_j \in E(\phi)$ for a clause $c_j \in C$ and variable $x_i \in X$ if and only if x_i occurs in c_j .
- $x_i x_{i+1} \in E(\phi)$ for all $1 \leq i < n$.

A formula ϕ in 3-CNF is called *positive planar* if it is both positive and backbone planar. The problem of POSITIVE PLANAR 1-IN-3-SAT consists of deciding whether a given positive planar 3-CNF formula allows a truth assignment such that in each clause, exactly one literal is true.

Theorem 1. *It is NP-complete even to decide whether an efficient guard set exists for a given polygon.*

Proof. Let $G(V, E)$ be a planar embedding of an instance of the POSITIVE PLANAR 1-IN-3-SAT problem. Corresponding to $G(V, E)$, we construct a polygon P such that an efficient guard set exists for P if and only if the instance of POSITIVE PLANAR 1-IN-3-SAT is satisfiable.

Consider a formula $\phi = (C, X)$ with a planar embedding as shown in Fig. 1.

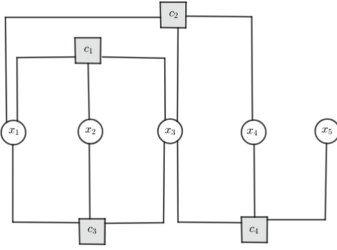


Fig. 1. Planar embedding of POSITIVE PLANAR 1-IN-3-SAT

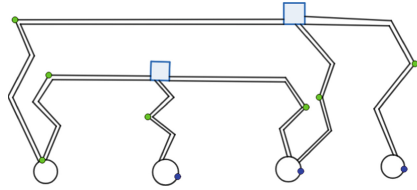


Fig. 2. Connecting clauses and variables - the squares represent clauses and the circles represent variables

Given a planar embedding, corresponding to each clause, we construct a polygon of a specific type and corresponding to each variable we construct another type of polygon. The edges that go from clauses to variables are also replaced by zigzag polygons connecting the clause polygon to variable polygon. Figure 2 depicts the replacement of the edges. Figures 3 and 4 depict clause and variable polygons.

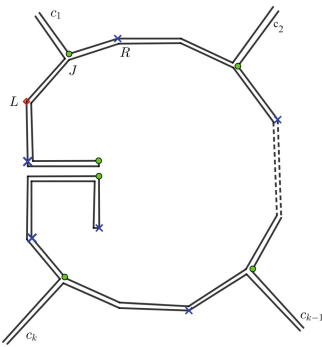


Fig. 3. Variable polygon

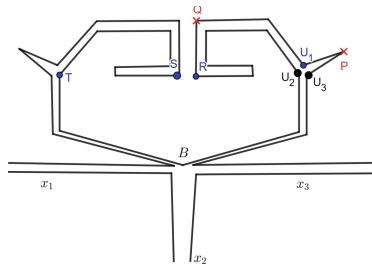


Fig. 4. Clause polygon

We first describe the variable and clause polygons. Let the joining of two tube-like structures be called a *junction*, i.e., wherever there is a bend in the

polygon. Each junction has a set of vertices. Note that the structure of the polygon ensures the visibility regions of all the vertices belonging to any given junction are the same. Hence, if a guard has to be placed at a junction, the guard can be placed at any vertex belonging to the junction. For simplicity, one vertex is marked in the figures.

As defined, each clause has three variables. And each variable occurs in at least one clause. The polygon for a clause is fixed and is as shown in Fig. 4. The three extensions at the bottom of the clause polygon, marked as x_1 , x_2 , and x_3 , are the zigzag polygons which connect to the corresponding variable polygons. Each of these zigzag polygons has a constant number of vertices. In the latter part of the proof, we show how the variable polygons are connected to clause polygons through these open tubes. The clause polygon has two additional spikes, which also need to be guarded.

As already mentioned, a variable can belong to any number of clauses as long as the planarity condition is satisfied. The structure of a variable polygon is defined by the number of clauses the corresponding variable belongs to. If a variable belongs to k number of clauses, a tube polygon with $3k$ sides is used as a skeleton for the constructing the corresponding variable polygon, where every third junction is a *clause junction*, i.e., a tube that connects to the corresponding clause polygon emerges from this junction. Finally, one edge of the polygon is removed and replaced with an L -shaped tube structure and a horizontal tube structure as shown in Fig. 3. Suppose we have a POSITIVE PLANAR 1-IN-3-SAT problem with n variables and m clauses. It is easy to see that the constructed polygon \mathcal{P} has $O(n + m)$ vertices. The proof of Theorem 1 follows from the following lemma.

Lemma 1. *The constructed polygon \mathcal{P} has an efficient guard set if and only if the formula corresponding to $G(V, E)$ is satisfiable.*

4 FPT Algorithm for Terrains

In this section, we consider the EFFICIENT GUARDING problem for 1.5 dimensional terrains. A 1.5 dimensional terrain is an x -monotone chain of line segments i.e., if the set of n vertices in the order of their x -coordinates is $\{x_1, \dots, x_n\}$, then there exists an edge $x_i x_{i+1}$ for all $1 \leq i < n$. Two vertices see each other if the line segment connecting them lies completely above the polygonal chain. Consider the visibility graph G of the terrain T . We have already mentioned that the EFFICIENT GUARDING problem on T is equivalent to the EFFICIENT DOMINATION problem on G .

Onion Peeling Number of a Terrain: Onion peeling number is the number of convex layers of the polygon. For terrains, onion peeling number is defined as the number of upper convex hulls of the terrain. Khodakarami et al. [19] have proved the following result for terrains:

Lemma 2 ([19]). *Let depth of the onion peeling be bounded by k , then the treewidth of the 1.5D terrain visibility graph is bounded by $2k$.*

We now present an FPT algorithm for EFFICIENT DOMINATION parameterized by treewidth. By Lemma 2, this also proves that the EFFICIENT GUARDING of terrains parameterized by the onion peeling number is in FPT. Moreover, this gives an efficient algorithm for EFFICIENT GUARDING of polygons with bounded treewidth visibility graph.

Our algorithm uses techniques similar to the algorithm for minimum dominating set problem, given in [1]. We use a refined variant of the nice tree decomposition. The nice tree decomposition includes a new type of node called *introduce edge node*, which is labeled with an edge uv such that $u, v \in X_t$ and with exactly one child t' such that $X_t = X_{t'}$; uv is said to be *introduced* at node t . This variant of tree decomposition still has $O(\tau \cdot n)$ nodes, where τ is the treewidth of the graph G . With each node t of the tree decomposition we associate a subgraph G_t of G defined as: $G_t = (V_t, E_t = \{e : e \text{ is introduced in the subtree rooted at } t\})$. Here, V_t is the union of all bags present in the subtree rooted at t .

We define subproblems on t for the graph G_t . We consider a coloring of the vertices in each bag X_t using colors from $\{\text{black}, \text{white}, \text{grey}\}$. Intuitively, the meaning of each color is as follows:

Black: every black vertex is part of the dominating set and is also efficiently dominated in the partial solution for G_t , i.e, it is dominated by itself.

White: a white vertex is not a part of the dominating set in the partial solution for G_t , but is efficiently dominated in the partial solution by a black vertex.

Grey: a grey vertex is not a part of the dominating set and is not dominated in the partial solution. It is dominated in the tree above the bag t .

The reason we introduce grey vertices is that some vertices of a bag can be dominated by vertices or edges not introduced so far. Thus, we consider subproblems where some vertices of the bag are not efficiently dominated in G_t and will be efficiently dominated by vertices or edges introduced later. We stress the fact that it is strictly forbidden to dominate a grey vertex in G_t .

For a node t , there are $3^{|X_t|}$ colorings of X_t . A coloring f is valid if

- Every vertex v which is either black or white has exactly one neighbor belonging to the dominating set, i.e., belonging to the black partition. Note that for a black vertex, the neighbor dominating it is itself since we consider closed neighborhood.
- No vertex v in the grey partition has a black neighbor in G_t .

For a given node t and a coloring f of X_t , we define by $d[t, f]$ of G_t as follows. If f is valid then $d[t, f] = s$ where $|f^{-1}(\text{black})| = s$, otherwise $d[t, f] = \infty$. Note that the value of $d[r, \phi]$ where r is the root of the tree decomposition denotes the minimum size of an efficient dominating set if it exists and ∞ otherwise. This is the case where $G = G_r$ and there is no corresponding partitioning for the root node.

We introduce additional notations. For a subset $X \subseteq V(G)$, consider a coloring $f : X \rightarrow \{\text{white}, \text{black}, \text{grey}\}$. For a vertex $v \in V(G)$ and a color $\alpha \in$

$\{white, black, grey\}$ we define a new coloring $f_{v \rightarrow \alpha} : X \cup \{v\} \rightarrow \{white, black, grey\}$ as follows:

$$f_{v \rightarrow \alpha} = \begin{cases} f(x) & \text{when } x \neq v \\ \alpha & \text{when } x = v \end{cases} \quad (1)$$

For a coloring f of X and $Y \subseteq X$, we use $f|_Y$ to denote the restriction of f to Y .

We now proceed to define the recursive formulas for the values of d .

Leaf Node. For a leaf node t , we have $X_t = \phi$. Hence, only the empty coloring is possible, and hence $d[t, \phi] = 0$.

Introduce Vertex Node. Let t be the introduce vertex node with a child t' such that $X_t = X_{t'} \cup \{v\}$ for some $v \notin X_{t'}$. The vertex v is isolated in G_t since no edges incident to v are introduced in this node. An isolated white vertex cannot be introduced, since the coloring would then be invalid. Hence, the vertex v is either added to the black or grey partitions by assigning colors accordingly. In the case where the vertex is added to the dominating set, the vertex is efficiently dominated by itself.

$$d[t, f] = \begin{cases} d[t', f|_{X'}] + 1 & \text{if } f(v) = black \\ d[t', f|_{X'}] & \text{if } f(v) = grey \\ \infty & \text{otherwise} \end{cases} \quad (2)$$

Introduce Edge Node. Let t be an introduce edge node labeled with an edge uv and let t' be the child of it. The edge uv can help in efficiently dominating vertices. However, if u was a black vertex before edge uv is introduced, then v will be dominated by u when the edge uv is introduced, therefore the coloring will not remain valid if v was black or white. If v was grey, it is now dominated by u and hence the coloring is valid if v has color white.

$$d[t, f] = \begin{cases} d[t', f_{v \rightarrow grey}] & \text{if } (f(u), f(v)) = (black, white) \\ d[t', f_{u \rightarrow grey}] & \text{if } (f(u), f(v)) = (white, black) \\ \infty & \text{if } (f(u), f(v)) \in \{(grey, black), (black, grey), (black, black)\} \\ d[t', f] & \text{otherwise} \end{cases} \quad (3)$$

Forget Node. Let t be a forget node with child t' such that $X_t = X_{t'} \setminus \{w\}$ for some $w \in X_{t'}$. Since the vertex w is not seen again in any node above t , if w is to be efficiently dominated in G_t then w needs to be efficiently dominated in $G_{t'}$ and hence w must be in either black or white. This gives the following recursive formula.

$$d[t, f] = \min(d[t', f_{w \rightarrow white}], d[t', f_{w \rightarrow black}]) \quad (4)$$

Join Node. Let t be the join node with children t_1 and t_2 . We know that $X_t = X_{t_1} = X_{t_2}$. We say that coloring f_1 of X_{t_1} and f_2 of X_{t_2} are *consistent* with a coloring f of X_t if for every $v \in X_t$ the following conditions hold:

- $f(v) = \textit{grey}$ if and only if $f_1(v) = f_2(v) = \textit{grey}$.
- $f(v) = \textit{black}$ if and only if $f_1(v) = f_2(v) = \textit{black}$.
- $f(v) = \textit{white}$ if and only if $(f_1(v), f_2(v)) \in \{(\textit{white}, \textit{grey}), (\textit{grey}, \textit{white})\}$.

Other possible colorings of f_1 and f_2 are not consistent. Consider the case where $(f_1(v), f_2(v)) = (\textit{white}, \textit{white})$. In this case, we can infer that the dominating vertices of the vertex v are different in G_{t_1} and G_{t_2} . This stems from the definition of introduce edge node. If the vertex v is dominated by a vertex u_1 in G_{t_1} , the vertex u_1 must be forgotten in the subtree rooted at t_1 because vertex u_1 can dominate vertex v through an introduce edge node only and the vertex u_1 is forgotten after the introduce edge node. Hence, the vertex v cannot be dominated by the same vertex u_1 in G_{t_2} also, since the vertex u_1 cannot be present in G_{t_2} as per the definition of tree decomposition (all the nodes containing a vertex u must form a connected subtree and the vertex u_1 was forgotten in the subtree rooted at t_2). Furthermore, all the remaining cases where $f_1(v) \in \{\textit{white}, \textit{grey}\}$ and $f_2(v) \in \{\textit{black}\}$, and the symmetric cases are not consistent since the vertex v belongs to the dominating set in one child node and does not belong to the dominating set in the other child node.

Now, considering all the colorings f_1, f_2 which are consistent, we define the recursive formula for the join node.

$$d[t, f] = \min_{f_1, f_2} (d[t_1, f_1] + d[t_2, f_2] - f^{-1}(\textit{black})) \quad (5)$$

where f_1 and f_2 are colorings consistent with f .

We have described the recursive formulas for the values of d . We analyze the running time of the algorithm. Clearly, the time needed to process a leaf node, introduce edge/vertex node or forget node is at most $3^\tau \cdot \tau^{O(1)}$, assuming adjacency queries can be carried out in $O(\tau)$ time. However, computing the values at a join node is more time-consuming. Note that if a pair f_1, f_2 is consistent with f , then for every $v \in X_t$ we have

$$(f(v), f_2(v), f_2(v)) \in \{(\textit{grey}, \textit{grey}, \textit{grey}), (\textit{white}, \textit{grey}, \textit{white}), (\textit{white}, \textit{white}, \textit{grey}), (\textit{black}, \textit{black}, \textit{black})\}$$

It follows that there are exactly $4^{|X_t|}$ triples of colorings of (f, f_2, f_2) such that f_1 and f_2 are consistent with f . We iterate through all the valid triples, and for each configuration we check if f_1 and f_2 are both valid colorings in their respective bags and update the value of $d[t, f]$ accordingly. It follows that the total time spent at each join node is at most $(4^\tau \cdot \tau^{O(1)})$. Since the total number of nodes in the modified variant of the nice tree decomposition is $O(\tau \cdot n)$, we derive the following theorem.

Theorem 2. *Let G be an n vertex graph together with a nice tree decomposition of treewidth τ . EFFICIENT DOMINATION problem on G is in FPT when parameterized by τ , with a running time of $4^\tau \cdot \tau^{O(1)} \cdot n$.*

Corollary 1. EFFICIENT GUARDING on 1.5 dimensional terrains is fixed parameter tractable when parameterized by the onion peeling number.

5 Polynomial Time Algorithms

In this section, we discuss polynomial time algorithms to solve EFFICIENT GUARDING for two special type of polygons.

Tower Polygons: A tower polygon is a polygon with two reflex chains joined together at a convex vertex and bounded below by an edge.

Without loss of generality, we use the term *left (right) reflex chain* to denote the reflex chain that extend from the topmost point to the leftmost(rightmost) point. Let l_1, l_2, \dots, l_n denote the vertices of the left reflex chain and r_1, r_2, \dots, r_m denote the vertices of the right reflex chain, in decreasing order of y-co-ordinates. Note that l_1 and r_1 denote the same point and l_n and r_m form an edge. Except for the angles at vertices l_1, l_n and r_m , all other angles are reflex. The following lemma states some properties of visibility in a tower polygon.

Lemma 3. 1. Any vertex except l_1, l_n and r_m sees only two vertices (other than itself) from the same chain. l_1, l_n and r_m sees only one vertex from the same chain.

2. For a vertex v , the vertices seen by v on the opposite chain are consecutive.

For a vertex v , let $s(v)$ represent the first vertex in the opposite chain that is seen by v and $f(v)$ be the last vertex in the opposite chain that is seen by v .

Lemma 4. Let $i < j$ and $h < k$. Then if l_i sees r_k and r_h sees l_j , then l_i sees r_h .

If l_{n-1} sees r_m add l_{n-1} to a set R . Similarly add r_{m-1} to R if it sees l_n .

Lemma 5. Exactly one vertex from $\{l_n, r_m\} \cup R$ is part of the efficient guarding set.

Now we show that the EFFICIENT GUARDING problem on tower polygons can be solved using a dynamic programming algorithm. Let $EG(v)$ denote the size of the minimum sized efficient guard set that guards all the vertices upto $v+1$ in the same chain and upto $f(v)$ in the opposite chain and in addition, the guard set also contains v . It is clear from Lemma 5 that the solution for minimum efficient guarding of P is given by $\min_{v \in \{l_n, r_m\} \cup R} EG(v)$. We now give a recurrence formula for $EG(v)$.

For a given vertex l_i in the left chain, we define a set R_i as follows. Let $r_j = s(l_i)$. R_i contains the vertex l_{i-3} if $f(l_{i-3}) = r_{j-1}$. R_i contains the vertex r_{j-2} if $f(r_{j-2}) = l_{i-2}$.

$$EG(l_i) = \begin{cases} \infty, & \text{if } R_i = \emptyset \\ \min_{v \in R_i} EG(v) + 1, & \text{otherwise} \end{cases}$$

We now show the correctness of the recurrence formula. By the definition of $EG(l_i)$, l_i is part of the efficient guard set and hence l_{i-1} is guarded by l_i . The vertex l_{i-2} is not seen by l_i and it cannot be guarded by itself because that means l_{i-1} is not efficiently guarded. Therefore, l_{i-2} can be guarded only by l_{i-3} if it is to be guarded by a vertex in the left chain. Similarly r_j is guarded by l_i and r_{j-1} is not. r_{j-1} can be guarded by r_{j-2} or some vertex in the left chain. By Lemma 4, l_{i-2} cannot be guarded by a vertex in the right chain and r_{j-1} cannot be guarded by a vertex in the left chain, simultaneously. Also, it is not possible that l_{i-2} is guarded by l_{i-3} and r_{j-1} is uniquely guarded by some l_k where $k < i - 3$, since this implies $f(l_{i-3})$ lies above $f(l_k)$, which is not possible by Lemma 4. Therefore, both l_{i-2} and r_{j-1} should be guarded either by l_{i-3} or r_{j-2} , if their visibility region contains both l_{i-2} and r_{j-1} and no vertex which is already seen by l_i . This proves the correctness of the recurrence. Observe that the recurrence formula for $EG(r_i)$ is symmetrical.

For a given vertex v , $EG(v)$ can be computed in constant time if the values for vertices above v is already known. Since we compute the value for each vertex, the algorithm runs in $O(n)$ time.

Theorem 3. *When the visibility graph is given, the minimum efficient guarding problem for a tower polygon can be solved in $O(n)$ time.*

Spiral Polygons: Spiral polygons are polygons in which all the reflex vertices occur consecutively along the boundary. The following result follows from results proved in [8,14].

Theorem 4. *The minimum efficient guarding problem on spiral polygons can be solved in polynomial time.*

References

1. Alber, J., Bodlaender, H.L., Fernau, H., Kloks, T., Niedermeier, R.: Fixed parameter algorithms for dominating set and related problems on planar graphs. *Algorithmica* **33**(4), 461–493 (2002)
2. Ashok, P., Fomin, F.V., Kolay, S., Saurabh, S., Zehavi, M.: Exact algorithms for terrain guarding. *ACM Trans. Algorithms (TALG)* **14**(2), 25 (2018)
3. Bange, D.W., Barkauskas, A.E., Slater, P.J.: Efficient dominating sets in graphs. *Appl. Discrete Math.* **189**, 189–199 (1988)
4. Bärtschi, A., Ghosh, S.K., Mihalák, M., Tschager, T., Widmayer, P.: Improved bounds for the conflict-free chromatic art gallery problem. In: *Proceedings of the Thirtieth Annual Symposium on Computational Geometry*, p. 144. ACM (2014)
5. Bärtschi, A., Suri, S.: Conflict-free chromatic art gallery coverage. *Algorithmica* **68**(1), 265–283 (2014)
6. Biggs, N.: Perfect codes in graphs. *J. Comb. Theory Ser. B* **15**(3), 289–296 (1973)
7. Bonnet, E., Miltzow, T.: Parameterized hardness of art gallery problems. In: *24th Annual European Symposium on Algorithms, ESA 2016, Aarhus, Denmark, 22–24 August 2016*, pp. 19:1–19:17 (2016)

8. Chang, G.J., Pandu Rangan, C., Coorg, S.R.: Weighted independent perfect domination on cocomparability graphs. In: Ng, K.W., Raghavan, P., Balasubramanian, N.V., Chin, F.Y.L. (eds.) ISAAC 1993. LNCS, vol. 762, pp. 506–514. Springer, Heidelberg (1993). https://doi.org/10.1007/3-540-57568-5_282
9. Chvatal, V.: A combinatorial theorem in plane geometry. *J. Comb. Theory Ser. B* **18**(1), 39–41 (1975)
10. Cygan, M., et al.: *Parameterized Algorithms*, vol. 3. Springer, Cham (2015). <https://doi.org/10.1007/978-3-319-21275-3>
11. Downey, R.G., Fellows, M.R.: *Parameterized Complexity*. Springer, New York (2012)
12. Eidenbenz, S., Stamm, C., Widmayer, P.: Inapproximability of some art gallery problems. In: CCCG, pp. 64–65 (1998)
13. Erickson, L.H., LaValle, S.M.: An art gallery approach to ensuring that landmarks are distinguishable. In: *Robotics: Science and Systems*, vol. 7, pp. 81–88 (2012)
14. Everett, H., Corneil, D.G.: Recognizing visibility graphs of spiral polygons. *J. Algorithms* **11**(1), 1–26 (1990)
15. Fekete, S.P., Friedrichs, S., Hemmer, M.: Complexity of the general chromatic art gallery problem. arXiv preprint [arXiv:1403.2972](https://arxiv.org/abs/1403.2972) (2014)
16. Ghosh, S.K.: *Visibility Algorithms in the Plane*. Cambridge University Press, Cambridge (2007)
17. Ghosh, S.K.: Approximation algorithms for art gallery problems in polygons. *Discrete Appl. Math.* **158**(6), 718–722 (2010)
18. Gilbers, A., Klein, R.: A new upper bound for the VC-dimension of visibility regions. *Comput. Geom.* **47**(1), 61–74 (2014)
19. Khodakarami, F., Didehvar, F., Mohades, A.: A fixed-parameter algorithm for guarding 1.5D terrains. *Theor. Comput. Sci.* **595**, 130–142 (2015)
20. Khodakarami, F., Didehvar, F., Mohades, A.: 1.5D terrain guarding problem parameterized by guard range. *Theor. Comput. Sci.* **661**, 65–69 (2017)
21. King, J., Kirkpatrick, D.: Improved approximation for guarding simple galleries from the perimeter. *Discrete Comput. Geom.* **46**(2), 252–269 (2011)
22. King, J., Krohn, E.: Terrain guarding is NP-hard. *SIAM J. Comput.* **40**(5), 1316–1339 (2011)
23. Krohn, E., Nilsson, B.J.: The complexity of guarding monotone polygons (2012)
24. Lee, D., Lin, A.: Computational complexity of art gallery problems. *IEEE Trans. Inf. Theory* **32**(2), 276–282 (1986)
25. Liang, Y.D., Lu, C.L., Tang, C.Y.: Efficient domination on permutation graphs and trapezoid graphs. In: Jiang, T., Lee, D.T. (eds.) COCOON 1997. LNCS, vol. 1276, pp. 232–241. Springer, Heidelberg (1997). <https://doi.org/10.1007/BFb0045090>
26. Lu, C.L., Tang, C.Y.: Weighted efficient domination problem on some perfect graphs. *Discrete Appl. Math.* **117**(1–3), 163–182 (2002)
27. Matoušek, J., Sharir, M., Welzl, E.: A subexponential bound for linear programming. *Algorithmica* **16**(4–5), 498–516 (1996)
28. Mulzer, W., Rote, G.: Minimum-weight triangulation is NP-hard. *J. ACM (JACM)* **55**(2), 11 (2008)
29. O’Rourke, J.: *Art Gallery Theorems and Algorithms*, vol. 57. Oxford University Press, Oxford (1987)
30. Schuchardt, D., Hecker, H.-D.: Two NP-hard art-gallery problems for orthopolygons. *Math. Logic Q.* **41**(2), 261–267 (1995)



Graph Orientation with Edge Modifications

Yuichi Asahiro¹(✉), Jesper Jansson², Eiji Miyano³, Hiroataka Ono⁴,
and Sandhya T. P.²

¹ Department of Information Science, Kyushu Sangyo University, Fukuoka, Japan
asahiro@is.kyusan-u.ac.jp

² The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong
jesper.jansson@polyu.edu.hk, csstp@comp.polyu.edu.hk

³ Department of Artificial Intelligence, Kyushu Institute of Technology, Iizuka,
Fukuoka, Japan
miyano@ces.kyutech.ac.jp

⁴ Graduate School of Informatics, Nagoya University, Nagoya, Japan
ono@i.nagoya-u.ac.jp

Abstract. The goal of an *outdegree-constrained edge-modification problem* is to find a spanning subgraph or supergraph H of an input undirected graph G such that either: (*Type I*) the number of edges in H is minimized or maximized and H can be oriented to satisfy some specified constraints on the vertices' resulting outdegrees; or: (*Type II*) the maximum or minimum outdegree of all vertices under an optimal orientation of H is minimum or maximum among all subgraphs or supergraphs of G that can be constructed by deleting or inserting a fixed number of edges. This paper introduces eight new outdegree-constrained edge-modification problems related to load balancing called (*Type I*) MIN-DEL-MAX, MIN-INS-MIN, MAX-INS-MAX, and MAX-DEL-MIN and (*Type II*) p -DEL-MAX, p -INS-MIN, p -INS-MAX, and p -DEL-MIN. We first present a framework that provides algorithms for solving all eight problems in polynomial time on unweighted graphs. Next we investigate the inapproximability of the edge-weighted versions of the problems, and design polynomial-time algorithms for six of the problems on edge-weighted trees.

Keywords: Graph orientation · Maximum flow ·
Computational complexity · Inapproximability · Greedy algorithm ·
Load balancing

1 Introduction

Graph modification problems are fundamental in graph theory and arise in many theoretical and practical settings, including computational biology [15] and machine learning [6]. Given a weighted or unweighted graph $G = (V, E)$ and a graph property Π , the general objective is to transform the graph G into a

graph G' satisfying property Π by applying a shortest sequence of graph modification operations. There are two main types of graph modification problems: *vertex-modification problems* and *edge-modification problems*. In the former, one is allowed to add or remove vertices from the graph, while in the latter, the goal is typically to find a spanning subgraph or supergraph of G satisfying property Π .

A special case of edge-modification problems is when the property Π depends on the vertices' degrees. Such *degree-constrained edge-modification* problems are very general and include many natural problems such as the MAXIMUM WEIGHT PERFECT MATCHING, MAXIMUM r -FACTOR, LONGEST CYCLE, and GENERAL FACTOR problems. Indeed, one can regard MAXIMUM WEIGHT PERFECT MATCHING (or MAXIMUM r -FACTOR) as the problem of finding a subgraph G' of G such that: (i) the degree of every vertex in $V(G')$ is one (or r .); and (ii) the total weight of the deleted edges is minimized. Similarly, LONGEST CYCLE is equivalent to the problem of finding a subgraph G' such that: (i) the degree of every vertex in $V(G')$ is two; (ii) G' is connected; and (iii) the total weight of the deleted edges is minimized. GENERAL FACTOR asks if it is possible to delete edges from G so that the resulting graph G' is connected and every vertex v in G' has degree equal to a number belonging to a specified set $K(v)$. Many of these problems are NP-hard; e.g., LONGEST CYCLE as well as GENERAL FACTOR are NP-hard even for unweighted graphs. Therefore, it is important to identify special cases of them that can be solved efficiently. One such special case of GENERAL FACTOR is the new problem MIN-DEL-MAX, introduced below.

An *orientation* of an undirected graph is an assignment of a direction to each of its edges. By an *outdegree-constrained edge-modification* problem, we mean an edge-modification problem where the solution is required to admit an orientation in which the vertices' outdegrees satisfy some specified constraints. This paper introduces eight new outdegree-constrained edge-modification problems including MIN-DEL-MAX. Besides the fact that MIN-DEL-MAX is a special case of GENERAL FACTOR as mentioned in the above, MIN-DEL-MAX and the other seven problems are related to load balancing which is also an important research topic. Here we explain about the relation between MIN-DEL-MAX and a load balancing problem as an example: Suppose there is a set of jobs to be completed, each job can be processed by exactly one of two specified machines, assuming that for any pair of machines at most one job is imposed, and we initially want to assign each job to a machine while minimizing the maximum load on the machines. This situation is represented by a graph with vertices interpreted as machines and edges interpreted as jobs. An orientation of such a graph corresponds to an assignment of jobs, where the start vertex (machine) of a directed edge processes the edge (job). Unfortunately, after choosing one assignment, it turns out that the maximum load is too high, so we have to give up trying to complete all of the jobs. Instead, we compute the fewest jobs to abandon in order to decrease the resulting maximum load to within some reasonable amount. This procedure corresponds to finding a smallest set of deleting edges in the above mentioned graph. The other seven problems have similar interpretations.

We first provide algorithms for solving all of the eight new problems in polynomial time on unweighted graphs. We then prove that their generalizations to

edge-weighted graphs cannot be approximated within a ratio of $\rho(n)$, for instance, in polynomial time unless $P = NP$, where n is the number of vertices in the input graph and $\rho(n) \geq 1$ is any polynomial-time computable function. This inapproximability holds even for planar bipartite graphs. Finally, as a tractable subclass of the planar bipartite graphs, we consider the problems on edge-weighted trees.

1.1 Problem Definitions

Let $G = (V, E)$ be a simple, undirected graph, where V and E are the set of vertices and the set of edges, respectively. For any $u, v \in V$, the undirected edge with endpoints in u and v is denoted by $\{u, v\}$, and the directed edge from u to v is denoted by (u, v) . For G , its complement (V, \bar{E}) is denoted by G^c , where $\bar{E} = \{\{u, v\} \mid u, v \in V, u \neq v, \{u, v\} \notin E\}$. An *orientation* Λ of G is an assignment of a direction to each edge in E , i.e., every $\{u, v\} \in E$ is set to either (u, v) or (v, u) . (Equivalently, Λ is a set of directed edges that consists of exactly one of the two directed edges (u, v) and (v, u) for every $\{u, v\} \in E$.) Let $\Lambda(G)$ denote the directed graph (V, Λ) . For any $v \in V$ and a fixed orientation Λ of G , define $d^+(v)$ as the outdegree of v under Λ . Finally, let $\Gamma(G)$ be the set of all orientations of G .

We now define the first four new graph orientation problems. Each of them takes as input a simple, undirected graph $G = (V, E)$ and a positive integer k' .

- MIN-DEL-MAX: (Assumes w.l.o.g. that $k' \leq \min_{\Lambda \in \Gamma(G)} \max_{u \in V} d^+(u)$.) Find the minimum number of edges whose deletion results in a graph G' with $\min_{\Lambda \in \Gamma(G')} \max_{u \in V} d^+(u) \leq k'$.
- MIN-INS-MIN: (Assumes w.l.o.g. that $k' \geq \max_{\Lambda \in \Gamma(G)} \min_{u \in V} d^+(u)$.) Find the minimum number of edges whose addition results in a simple graph G' with $\max_{\Lambda \in \Gamma(G')} \min_{u \in V} d^+(u) \geq k'$.
- MAX-INS-MAX: (Assumes w.l.o.g. that $k' \geq \min_{\Lambda \in \Gamma(G)} \max_{u \in V} d^+(u)$.) Find the maximum number of edges whose addition results in a simple graph G' with $\min_{\Lambda \in \Gamma(G')} \max_{u \in V} d^+(u) \leq k'$.
- MAX-DEL-MIN: (Assumes w.l.o.g. that $k' \leq \max_{\Lambda \in \Gamma(G)} \min_{u \in V} d^+(u)$.) Find the maximum number of edges whose deletion results in a graph G' with $\max_{\Lambda \in \Gamma(G')} \min_{u \in V} d^+(u) \geq k'$.

Observe that in the problems MIN-INS-MIN and MAX-INS-MAX, the resulting graphs must be simple. The above four problems optimize the number of edges to delete or insert; we also define four related problems that take as input a simple, undirected graph $G = (V, E)$ and whose objectives are to optimize the maximum/minimum outdegree for a fixed integer p representing the number of deleted/inserted edges:

- p -DEL-MAX: Find the smallest possible value of $\min_{\Lambda \in \Gamma(G(V, E \setminus E'))} \max_{u \in V} d^+(u)$ taken over all $E' \subseteq E$ with $|E'| = p$.
- p -INS-MAX: Find the smallest possible value of $\min_{\Lambda \in \Gamma(G(V, E \cup E'))} \max_{u \in V} d^+(u)$ taken over all $E' \subseteq E(G^c)$ with $|E'| = p$.
- p -INS-MIN: Find the largest possible value of $\max_{\Lambda \in \Gamma(G(V, E \cup E'))} \min_{u \in V} d^+(u)$ taken over all $E' \subseteq E(G^c)$ with $|E'| = p$.
- p -DEL-MIN: Find the largest possible value of $\max_{\Lambda \in \Gamma(G(V, E \setminus E'))} \min_{u \in V} d^+(u)$ taken over all $E' \subseteq E$ with $|E'| = p$.

Throughout the paper, let $n = |V|$ and $m = |E|$ for any given instance of the above eight problems. Δ is the (unweighted) maximum degree taken over all vertices in the input G . Any algorithm ALG is called a σ -approximation algorithm if the following inequality holds for every input graph G : $\max \left\{ \frac{\#\text{ALG}(G)}{\#\text{OPT}(G)}, \frac{\#\text{OPT}(G)}{\#\text{ALG}(G)} \right\} \leq \sigma$, where $\#\text{ALG}(G)$ and $\#\text{OPT}(G)$ are the number of deleted (or inserted) edges by ALG and an optimal algorithm, respectively.

1.2 Related Work

To compute $\min_{\Lambda \in \Gamma(G)} \max_{u \in V} d^+(u)$ for an input undirected, unweighted graph G is the MINIMUM MAXIMUM OUTDEGREE PROBLEM (MINMAXO), previously studied in [4, 7, 8, 10, 16]. MINMAXO can be solved in linear time for planar graphs [10] and in polynomial time for general graphs [16]. The problem of computing $\max_{\Lambda \in \Gamma(G)} \min_{u \in V} d^+(u)$, referred to as MAXMINO in [2], can also be solved in polynomial time (Theorem 8 in [2]). (This is why the input k' to MIN-DEL-MAX, MIN-INS-MIN, MAX-INS-MAX, and MAX-DEL-MIN can be assumed w.l.o.g. to satisfy $k' \leq \min_{\Lambda \in \Gamma(G)} \max_{u \in V} d^+(u)$ or $k' \geq \max_{\Lambda \in \Gamma(G)} \min_{u \in V} d^+(u)$.)

When generalized to edge-weighted graphs, both MINMAXO and MAXMINO become NP-hard [2, 4].

Theorem 8 in [2] states that MAXMINO is solvable in $O(m^{3/2} \log m \log^2 \Delta)$ time for unweighted graphs, which directly gives:

Theorem 1. *MAX-DEL-MIN can be solved in $O(nm^{3/2} \log m \log^2 \Delta)$ time.*

Proof. First compute an orientation by which the minimum outdegree has value $\max_{\Lambda \in \Gamma(G)} \min_{u \in V} d^+(u) (\geq k')$ using the algorithm for MAXMINO from [2], and obtain a directed graph. Then delete arbitrary $d^+(v) - k'$ outgoing edges for each vertex v in the directed graph to get a directed graph G' with $d^+(v) = k'$ for every $v \in V$. This deletion of edges only needs linear time since we can delete arbitrary set of outgoing edges for each vertex. The number of deleted edges is the maximum

Table 1. The computational complexity of the algorithms in Sect. 2 for unweighted graphs. For edge-weighted graphs all these problems are intractable, shown in Sect. 3.

Problem	Time complexity	Reference
MIN-DEL-MAX	$O(m^2 \log n)$	Theorem 3
MIN-INS-MIN	$O(n^4 \log n)$	Theorem 4
MAX-INS-MAX	$O(n^4 \log n)$	Theorem 4
MAX-DEL-MIN	$\min\{O(nm^{3/2} \log m \log^2 \Delta), O(m^2 \log n)\}$	Theorems 1 and 3
p -DEL-MAX	$O(m^2 \log n)$	Theorem 3
p -INS-MIN	$O(n^4 \log n)$	Theorem 4
p -INS-MAX	$O(n^4 \log n)$	Theorem 4
p -DEL-MIN	$O(m^2 \log n)$	Theorem 3

possible: Since every vertex has outdegree k' , the number of the directed edges is nk' in the graph, and so deleting any more edges would result in some vertex having outdegree strictly less than k' by the pigeonhole principle. Thus MAX-DEL-MIN can be solved in $O(nm^{3/2} \log m \log^2 \Delta)$ time. \square

A variant of MINMAXO in which one may perform p *split operations* on the vertices (corresponding to adding p extra machines in the load balancing setting described above) before orienting the edges was studied in [1]. That problem seems harder than the eight problems studied here, as it is NP-hard even for unweighted graphs when p is unbounded [1].

1.3 Our Contributions and Organization of the Paper

Section 2 presents polynomial-time algorithms for the new problems on unweighted graphs. See Table 1 for a summary of their computational complexity. In Sect. 3 we develop polynomial-time algorithms for six of the eight problems on edge-weighted trees, and prove the polynomial-time inapproximability for planar bipartite edge-weighted graphs for all eight problems. Finally, we conclude the paper in Sect. 4. Due to space constraints, some proofs are deferred to the full version of this paper.

2 Unweighted Graphs

In this section, we present polynomial-time algorithms for the unweighted versions of the new problems. Rather than developing a separate algorithm for each problem, our strategy is to give a unified framework from which each of the eight algorithms follows as a special case. We take advantage of the problems' structural similarities by encoding the input graph G in all eight cases as a directed graph N_G , augmenting N_G with edge capacities as defined below to obtain a

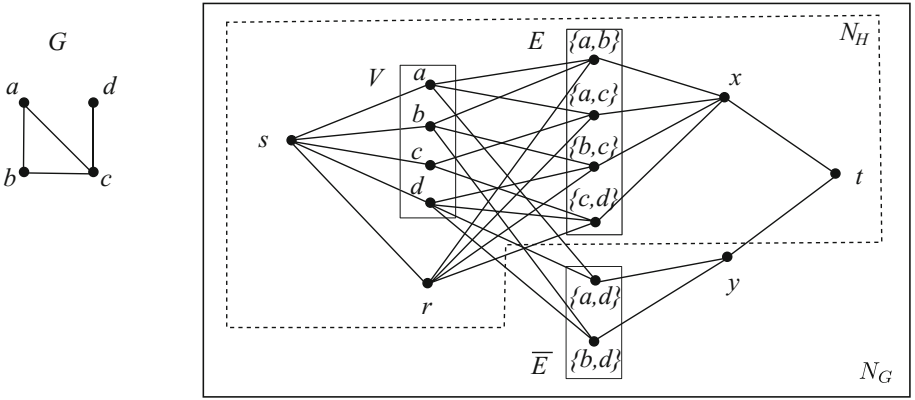


Fig. 1. A graph G and the directed graph N_G constructed from G . All edges are directed from left to right in N_G . The part surrounded by the dashed lines represents N_H .

flow network, and then using binary search together with a fast algorithm for computing maximum flows. N_G is the same for all problems; only the edge capacities in the flow network depend on which of the problems is being solved. The encoding used here is an extension of the one in [4]; to be precise, the definition of N_G follows the basic construction in [4] and then adds auxiliary vertices and directed edges that can capture the deletion and insertion of edges in the input graph.

The formal definition of N_G is as follows. For any undirected graph $G(V, E)$, construct the directed graph $N_G = (V_G, E_G)$ with vertex set V_G and edge set E_G by defining: $\bar{E} = \{\{u, v\} \mid u, v \in V, u \neq v, \{u, v\} \notin E\}$, $V_G = V \cup E \cup \bar{E} \cup \{x, y, r, s, t\}$, $E_G = \{(s, v) \mid v \in V\} \cup \{(v_i, e), (v_j, e) \mid e = \{v_i, v_j\} \in E \cup \bar{E}\} \cup \{(r, e) \mid e \in E\} \cup \{(e, x) \mid e \in E\} \cup \{(e, y) \mid e \in \bar{E}\} \cup \{(s, r), (x, t), (y, t)\}$. Note that if $e = \{u, v\} \in E$ (or \bar{E}), then N_G contains two directed edges (u, e) and (v, e) for e in the vertex subset E (or \bar{E}) of N_G . Note that the vertex r and the set of vertices in \bar{E} capture deletion and insertion of edges respectively, mentioned in the previous paragraph. See Fig. 1 for an illustration. We remark that maximum flows in suitably defined flow networks were previously used to solve some other graph orientation problems in [1–4]. The definition of N_G that we present here is more general.

Section 2.1 below explains how to use N_G to solve the problems MIN-DEL-MAX and p -DEL-MAX. Due to the space limitation, discussions for the other six problems are omitted. Then, Sect. 2.2 analyzes the time complexity of the obtained algorithms. To solve MIN-INS-MIN, p -INS-MIN, MAX-INS-MAX, and p -INS-MAX, we need to explicitly construct the entire directed graph N_G . However, for MIN-DEL-MAX, p -DEL-MAX, MAX-DEL-MIN, and p -DEL-MIN, the algorithms will only need the induced subgraph $N_H = N_G[V \cup E \cup \{s, r, x, t\}] = N_G \setminus (\bar{E} \cup \{y\})$ of N_G , and so these algorithms' running times will be lower when G is sparse.

2.1 Using N_G to Solve the Problems

First we focus on the problem MIN-DEL-MAX. For an undirected graph $G(V, E)$ with $k = \min_{\Lambda \in \Gamma(G)} \max_{u \in V} d^+(u)$ and a positive integer $k' \leq k$, construct the directed graph $N_H = N_G[V \cup E \cup \{s, r, x, t\}]$. For any positive integer p , let $N_H(p) = (V(N_H), E(N_H), \text{cap})$ be the flow network obtained by augmenting N_H with edge capacities cap , where:

$$\text{cap}(a) = \begin{cases} k', & \text{if } a = (s, v) \text{ for } v \in V \\ p, & \text{if } a = (s, r) \\ |E|, & \text{if } a = (x, t) \\ 1, & \text{otherwise} \end{cases}$$

The following lemma relates the size of the maximum flow in $N_H(p)$ to the number of deleted edges from G .

Lemma 2. *For an input graph $G(V, E)$ to MIN-DEL-MAX, there exists a flow in $N_H(p)$ with value $|E|$ if and only if there are exactly p edges in G whose deletion leaves a graph G' with $\min_{\Lambda \in \Gamma(G')} \max_{u \in V} d^+(u) \leq k'$.*

Proof. Suppose there exists a flow for the network $N_H(p)$ with value $|E|$. Since the edge capacities are integers, we can assume that the flow has nonnegative integral flows in each edge by the integrality theorem (see, e.g., [11]). Let j be the size of the flow through the vertex r and $j \leq p$. We consider those j edges of the form (r, e) , incident to r , through which the flow passes by. Let G' be the graph obtained by deleting those edges from G (If $j < p$, then delete $p - j$ arbitrary edges from G in order to ensure that exactly p edges are deleted from G). Then, construct an orientation of G' as follows. For every edge $f = (v, e)$ in $N_H(p)$ that contributes a unit of flow, where $v \in V$ and $e \in E$, orient the edge e away from v . Since every vertex $v \in V$ can have at most k' flow, $\max_{u \in V} d^+(u) \leq k'$ and hence $\min_{\Lambda \in \Gamma(G')} \max_{u \in V} d^+(u) \leq k'$.

Conversely, if there are p edges whose deletion leaves a graph G' with $\min_{\Lambda \in \Gamma(G')} \max_{u \in V} d^+(u) \leq k'$, then construct a flow with value $|E|$ as follows. Let $S = E \setminus E(G')$ and let Λ be a fixed orientation of G' that minimizes the maximum outdegree. For $e \in S$, send one unit of flow from s to r , r to e , e to x , and then x to t , and the total flow is p . If $e \notin S$, then send one unit of flow from s to v , v to e , e to x , and then x to t , where e is oriented away from v in Λ . For Λ , $\max_{u \in V} d^+(u) \leq k'$ and thus $d^+(u) \leq k'$, for every $u \in V$. Hence, through every vertex $v \in N_H(p)$ that corresponds to $v \in V$, at most k' units of flow pass, which is the capacity of the vertex v . In that way, every edge in G contributes one unit of flow and hence this particular flow of $N_H(p)$ has value $|E|$. \square

By Lemma 2, the minimum number of edges that can be deleted to get a graph G' with $\min_{\Lambda \in \Gamma(G')} \max_{u \in V} d^+(u) \leq k'$ is the same as the smallest value of p

```

1  $p_{\min} \leftarrow 0, p_{\max} \leftarrow |E|;$ 
2 repeat
3    $p \leftarrow \lfloor (p_{\min} + p_{\max})/2 \rfloor;$ 
4   Construct  $N_H(p)$  and let  $f$  be the value of the maximum flow of  $N_H(p)$  ;
5   if  $f = |E|$  then
6      $p_{\max} \leftarrow p;$ 
7   else
8      $p_{\min} \leftarrow p;$ 
9   end
10 until  $p_{\min} \geq p_{\max};$ 
11 Output  $p$  and halt;

```

Fig. 2. The algorithm for MIN-DEL-MAX on unweighted graphs

such that there exists a flow in $N_H(p)$ of value $|E|$. A binary search on p will give the minimum value of p for which there exists a flow of size $|E|$ in $N_H(p)$. The algorithm's pseudocode is listed in Fig. 2.

As for p -DEL-MAX, given an undirected graph $G(V, E)$ with $k = \min_{\Lambda \in \Gamma(G)} \max_{u \in V} d^+(u)$, build the directed graph $N_H = N_G[V \cup E \cup \{s, r, x, t\}]$. For any positive integer ℓ , let $N_H(\ell) = (V(N_H), E(N_H), \text{cap})$ be the flow network obtained by augmenting N_H with edge capacities cap , where:

$$\text{cap}(a) = \begin{cases} \ell, & \text{if } a = (s, v) \text{ for } v \in V \\ p, & \text{if } a = (s, r) \\ |E|, & \text{if } a = (x, t) \\ 1, & \text{otherwise} \end{cases}$$

By Lemma 2, the minimum value of the maximum outdegree is the same as the minimum value of ℓ such that there exists a flow of value $|E|$ in $N_H(\ell)$. Hence, by an algorithm similar to the one for MIN-DEL-MAX, for any ℓ , $0 \leq \ell \leq k$, we can check whether there is a flow of value $|E|$ and locate the smallest integer ℓ with this property by a binary search on ℓ .

2.2 Time Complexity of the Algorithms

Let $n = |V|$ and $m = |E|$ for a given graph $G = (V, E)$, and let $N = |V(N_H)|$ and $M = |E(N_H)|$. We note that for the directed graph N_H , $N = n + m + 4$ and $M = n + 4m + 2$. For MIN-DEL-MAX, the search for $p = O(m)$ can be carried out using binary search, which takes $O(\log p) = O(\log m) = O(\log n)$ time since $m = O(n^2)$. The maximum flow problem on $N_H(p)$ can be solved in $O(MN)$ time [14], so MIN-DEL-MAX can also be solved in $O(m^2 \log n)$ time. Since $\ell = O(n)$, in a similar way, we can analyze the time complexity of our algorithms for p -DEL-MAX, MAX-DEL-MIN, and p -DEL-MIN. Thus we have:

Theorem 3. *For unweighted graphs, MIN-DEL-MAX, p -DEL-MAX, MAX-DEL-MIN, and p -DEL-MIN can be solved in $O(m^2 \log n)$ time.*

As shown in Theorem 1, MAX-DEL-MIN can be solved in $O(nm^{3/2} \log m \log^2 \Delta)$ time. The above algorithm is faster than it when $m = \Omega(n^2)$ and so $\Delta = \Omega(n)$, that is, the input graph is very dense.

On a similar note, for the directed graph N_G , $N = |V_G| = \frac{n(n+1)}{2} + 5$ and $M = |E_G| = \frac{n(3n-1)}{2} + m + 3$. For MIN-INS-MIN, the search for $p = O(m)$ can be carried out using binary search and therefore takes $O(\log p) = O(\log m) = O(\log n)$ time. The maximum flow problem on $N_G(p)$ can be solved in $O(MN)$ time [14], so MIN-INS-MIN can also be solved in $O(n^4 \log n)$ time. Using $\ell = O(m)$, we can bound the time complexity of our algorithms for p -INS-MIN, MAX-INS-MAX, and p -INS-MAX in the same way. We obtain:

Theorem 4. *For unweighted graphs, MIN-INS-MIN, p -INS-MIN, MAX-INS-MAX, and p -INS-MAX can be solved in $O(n^4 \log n)$ time.*

3 Edge-Weighted Graphs

For the problems in this paper, we can define corresponding weighted versions. For the weighted versions of the problems, $d^+(v)$ for every vertex v and a fixed orientation represents the total weight of outgoing edges of v . Each of MIN-DEL-MAX, MAX-DEL-MIN, MIN-INS-MIN, and MAX-INS-MAX takes as input a simple undirected edge-weighted graph $G = (V, E, w)$ and the target maximum/minimum outdegree k' , where the function w assigns a positive integer (weight) to each edge. Then the objective of these problems is still to optimize the number of edges to delete/insert. (We do not optimize the total weight of deleted/inserted edges.) p -DEL-MAX, p -DEL-MIN, p -INS-MIN, and p -INS-MAX are similarly defined on edge-weighted graphs, where we need to delete/insert p edges for the problems.

3.1 Polynomial-Time Algorithms for Edge-Weighted Trees

Assuming $P \neq NP$, inapproximability of the weighted versions of the problems on planar bipartite graphs will be shown in Sect. 3.2. In this section, we design exact polynomial-time algorithms for some of the problems on weighted trees which is a representative subclass of planar bipartite graphs.

The important thing here is that we know the optimal costs for MINMAXO and MAXMINO on edge-weighted trees: For MINMAXO, the maximum outdegree of a vertex under any orientation is at least the maximum weight of the edges. Then, for MAXMINO, the minimum outdegree of a vertex under any orientation is 0, since there exist only $n - 1$ edges so that at least one of n vertices cannot have outgoing edges under any orientation. From this observation, an optimal orientation for edge-weighted trees is to orient all the edges towards an arbitrarily selected root vertex. Based on these observations, straightforward discussion gives the following lemma (the proof has been omitted here due to space limitations):

Lemma 5. *For edge-weighted trees, MAX-DEL-MIN and p -DEL-MIN can be solved in $O(n)$ time.*

In conjunction with the above lemma, we can show the next theorem. Although the above lemma is obtained as a direct consequence of the known results for MAXMINO, we need to design more complicated algorithms for MAX-DEL-MIN, p -DEL-MIN, MIN-INS-MIN, and p -INS-MIN (the proof has been omitted here due to space limitations).

Theorem 6. *For edge-weighted trees, MIN-DEL-MAX, p -DEL-MAX, MAX-DEL-MIN, p -DEL-MIN, and MIN-INS-MIN are solvable in $O(n)$ time. Also, p -INS-MIN is solvable in $O(n \log(w_{\max} \Delta))$ time, where w_{\max} is the maximum weight of edges and Δ is the maximum (unweighted) degree of vertices.*

3.2 Inapproximability for Edge-Weighted Planar Bipartite Graphs

MINMAXO is known to be NP-hard for edge-weighted planar bipartite graphs [5]. This implies the following inapproximability:

Theorem 7. *There is no polynomial-time $\rho(n)$ -approximation algorithm for MIN-DEL-MAX on edge-weighted planar bipartite graphs unless $P = NP$, where $\rho(n) \geq 1$ is any polynomial-time computable function.*

Proof. Suppose for the sake of obtaining a contradiction that there exists a polynomial-time $\rho(n)$ -approximation algorithm ALG for some polynomial-time computable function $\rho(n) > 1$ for MIN-DEL-MAX on edge-weighted planar bipartite graphs. Then, ALG can find an orientation in a given edge-weighted graph G in polynomial time such that the objective value $ALG(G)$ satisfies $OPT(G) \leq ALG(G) \leq \rho(n) \cdot OPT(G)$. Therefore, one can distinguish either $OPT(G) > 0$ or $OPT(G) = 0$ in polynomial time using ALG which admits the approximation ratio of $\rho(n)$, based on the observation that $ALG(G) > 0$ if and only if $OPT(G) > 0$. If $OPT(G) = 0$, then there is no need to remove any edge to make the outdegree of every vertex at most k , whereas if $OPT(G) \geq 1$, we need to remove at least one edge. This means that a decision version of MINMAXO with target value k can be solved in polynomial time, which contradicts the NP-hardness of MINMAXO on edge-weighted planar bipartite graphs. \square

The inapproximability bound 1.5 of MINMAXO for edge-weighted planar bipartite graphs gives the next theorem.

Theorem 8. *There is no polynomial-time 1.5-approximation algorithm for p -DEL-MAX on edge-weighted planar bipartite graphs unless $P = NP$.*

Proof. Consider an input planar bipartite graph G of MINMAXO. Add one new vertex u and one edge $\{u, v\}$ of weight $n \cdot w_{\max}$ to G for arbitrary $v \in V(G)$, where w_{\max} is the maximum weight of edges in G . Let this new graph be G' , where G' is also a planar bipartite graph. Observe that for 1-DEL-MAX, this new edge

should be deleted since otherwise the maximum outdegree is at least nw_{\max} , which is larger than the total weight of edges in G , and then we need to orient the edges in G optimally. Namely, if we can approximate 1-DEL-MAX within a ratio 1.5 for G' in polynomial-time, it also gives the answer to MINMAXO for G . This contradicts the inapproximability of MINMAXO, so that 1-DEL-MAX cannot be approximated within a ratio of 1.5 in polynomial-time. This discussion can be extended to the case $p \geq 2$ by adding p new vertices and p new edges of weight $n \cdot w_{\max}$ to G . The theorem follows. \square

The NP-hardness and the inapproximability bounds of MAXMINO and MINMAXO for edge-weighted planar bipartite graphs [2, 4] can be applied in the same way as Theorems 7 and 8 to obtain the following theorem (the proof has been omitted here due to space limitations):

Theorem 9. *There is no polynomial-time $\rho(n)(\rho(n), \rho(n), 2, 1.5, 2, \text{ resp.})$ -approximation algorithm for MIN-INS-MIN (MAX-INS-MAX, MAX-DEL-MIN, p -INS-MIN, p -INS-MAX, p -DEL-MIN, resp.) on edge-weighted planar bipartite graphs unless $P=NP$, where $\rho(n) \geq 1$ is any polynomial-time computable function.*

4 Concluding Remarks

We studied eight new graph orientation problems whose objective is to minimize/maximize the outdegree of the vertices after inserting/deleting edges, and presented polynomial-time algorithms for these problems on unweighted graphs. Also we showed the polynomial-time inapproximability for those problems on edge-weighted graphs, and polynomial-time algorithms for six of the problems were designed on edge-weighted trees. One of the further research topics is to study the complexity of MAX-INS-MAX and p -INS-MAX on edge-weighted trees. A natural generalization of MIN-DEL-MAX can be defined as follows:

Input: An unweighted graph $G = (V, E)$ and a mapping ρ that assigns to each vertex $v \in V$ an integer from $\{0, 1, \dots, \deg(v)\}$.

Goal: To find the minimum number of edges to delete to get a spanning subgraph H of G such that $d^+(u) \leq \rho(u)$ for every $u \in V$.

To solve this problem, we can first construct a directed graph N_G in the same way as in Sect. 2. Next, we augment edge capacities and costs to N_G to get a flow network by keeping the capacities and costs same as that of N_G for k' except the capacities of the directed edges of the form $\{(s, u) \mid u \in V\}$. The capacity of the directed edge (s, u) is defined to be $\rho(u)$ instead of k' , for every $u \in V$. Then we see that there exists a flow in N_G with value $|E|$ if and only if there are p edges in G whose deletion leaves a graph $d^+(u) \leq \rho(u)$, for every vertex $u \in V(G)$. In other words, the modified problem can be solved in polynomial time. In fact, both the modified problem and the original MIN-DEL-MAX have the same time complexity since the directed graphs that we construct in both cases are the same. We can generalize MIN-INS-MIN, MAX-INS-MAX, MAX-DEL-MIN in the same way and solve them in polynomial time as well.

The above problem is a special case of GENERAL FACTOR introduced by Lovász [12, 13], which is defined as

Input: An unweighted graph $G = (V, E)$ and a mapping K that assigns to each vertex $v \in V$ a set $K(v) \subseteq \{0, 1, \dots, \deg(v)\}$ of integers.

Goal: To check if there is a subgraph H of G s.t. $d_H(v) \in K(v)$ for every $v \in V$.

GENERAL FACTOR is a generalization of the factor problem, and NP-hard even for unweighted graphs [9]. Here the extension to the mapping from an integer to a set of integers makes the problem harder. We conjecture that the analogous generalizations to the problems in this paper are NP-hard as well.

Acknowledgments. This work was partially supported by JSPS KAKENHI Grant Numbers JP17K00016 and JP17K00024, and JST CREST JPMJR1402.

References

1. Asahiro, Y., Jansson, J., Miyano, E., Nikpey, H., Ono, H.: Graph orientation with splits. In: Lee, J., Rinaldi, G., Mahjoub, A.R. (eds.) ISCO 2018. LNCS, vol. 10856, pp. 52–63. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96151-4_5
2. Asahiro, Y., Jansson, J., Miyano, E., Ono, H.: Graph orientation to maximize the minimum weighted outdegree. *Int. J. Found. Comput. Sci.* **22**(3), 583–601 (2011)
3. Asahiro, Y., Jansson, J., Miyano, E., Ono, H.: Graph orientations optimizing the number of light or heavy vertices. *J. Graph Algorithms Appl.* **19**(1), 441–465 (2015)
4. Asahiro, Y., Jansson, J., Miyano, E., Ono, H., Zenmyo, K.: Approximation algorithms for the graph orientation minimizing the maximum weighted outdegree. *J. Comb. Optim.* **22**(1), 78–96 (2011)
5. Asahiro, Y., Miyano, E., Ono, H.: Graph classes and the complexity of the graph orientation minimizing the maximum weighted outdegree. *Discrete Appl. Math.* **159**, 498–508 (2011)
6. Bansal, N., Blum, A., Chawla, S.: Correlation clustering. *Mach. Learn.* **56**, 89–113 (2004)
7. Borradaile, G., Iglesias, J., Migler, T., Ochoa, A., Wilfong, G., Zhang, L.: Egalitarian graph orientations. *J. Graph Algorithms Appl.* **21**(4), 687–708 (2017)
8. Brodal, G.S., Fagerberg, R.: Dynamic representations of sparse graphs. In: Dehne, F., Sack, J.-R., Gupta, A., Tamassia, R. (eds.) WADS 1999. LNCS, vol. 1663, pp. 342–351. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48447-7_34
9. Cornuéjols, G.: General factors of graphs. *J. Comb. Theory Ser. B* **45**, 185–198 (1988)
10. Chrobak, M., Eppstein, D.: Planar orientations with low out-degree and compaction of adjacency matrices. *Theor. Comput. Sci.* **86**(2), 243–266 (1991)
11. Cormen, T., Leiserson, C., Rivest, R., Stein, C.: *Introduction to Algorithms*, 3rd edn. The MIT Press, Cambridge (2009)
12. Lovász, L.: The factorization of graphs. In: *Combinatorial Structures and Their Applications*, pp. 243–246 (1970)
13. Lovász, L.: The factorization of graphs II. *Acta Math. Acad. Sci. Hung.* **23**, 223–246 (1972)

14. Orlin, J.B.: Max flows in $O(nm)$ time, or better. In: Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC 2013), pp. 765–774. Association for Computing Machinery (ACM) (2013)
15. Sharan, R.: Graph modification problems and their applications to genomic research. Ph.D. thesis, School of Computer Science, Tel-Aviv University (2002)
16. Venkateswaran, V.: Minimizing maximum indegree. *Discrete Appl. Math.* **143**, 374–378 (2004)



Local Coloring: New Observations and New Reductions

Jie You^{1,2}(✉), Yixin Cao^{1,2}, and Jianxin Wang¹

¹ School of Computer Science and Engineering, Central South University, Changsha, China

youjie@csu.edu.cn, jxwang@mail.csu.edu.cn

² Department of Computing, Hong Kong Polytechnic University, Kowloon, Hong Kong, China

yixin.cao@polyu.edu.hk

Abstract. A k -coloring of a graph is an assignment of integers between 1 and k to vertices in the graph such that the endpoints of each edge receive different numbers. We study a local variation of the coloring problem, which imposes further requirements on every set of three vertices: We are not allowed to use two consecutive numbers for a path on three vertices, or three consecutive numbers for a cycle on three vertices. Given a graph G and a positive integer k , the local coloring problem asks for whether G admits a local k -coloring. We show that it cannot be solved in subexponential time, unless the Exponential Time Hypothesis fails, and a new reduction for the NP-hardness of this problem. Our structural observations behind these reductions are of independent interests. We close the paper with a short remark on local colorings of perfect graphs.

1 Introduction

Graph coloring, in its original form, asks for an assignment of colors for vertices of a graph, such that endpoints of every edge receive different colors. It is one of the most classic and best-known problems in graph theory. Since it nicely formulates the partition of objects (vertices) subject to conflict constraints (edges), it has many variations. In the local coloring introduced by Chartrand et al. [1, 2], there are additional requirements on every set of three vertices: We are not allowed to use two consecutive numbers for a path on three vertices, or three consecutive numbers for a cycle on three vertices. As usual, we use positive integers for colors. If all the integers are between 1 and k , then it is called a (local) k -coloring. The (local) *chromatic number* of a graph is the smallest k such that it has a (local) k -coloring.

Although it is an ostensibly naive extension of standard coloring, some subtleties of local coloring deserve attention. First of all, we are not counting the

Supported in part by the Hong Kong Research Grants Council (RGC) under grant 152261 and the National Natural Science Foundation of China (NSFC) under grants 61572414 and 61672536.

number of colors actually assigned (as in standard coloring). A local coloring usually leaves some gap in between; e.g., the local chromatic number for a triangle is 4, with a local coloring 1, 2, 4 or 1, 3, 4. This even led Chartrand et al. [1] to ask whether for all positive integer k , there are graphs of local chromatic number k of which every minimum local coloring uses all the k colors; see also [2].

In standard coloring, a vertex of degree one seldom concerns us: We can safely remove it, together with the edge incident to it, from the graph, unless it is the only edge of the graph, when the problem is completely trivial. In general, we can safely do away with all vertices of degree small than k when looking for a k -coloring, but this is not the case for local coloring. As shown in Fig. 1, even removing a vertex of degree-one from a graph may decrease its local chromatic number.

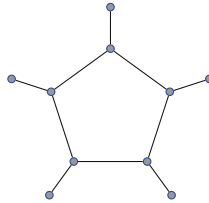


Fig. 1. The local chromatic number of this graph is 4. But if we delete any of the degree-one vertices, then its local chromatic number becomes 3, and the only degree-two vertex has to be colored by 2 in any local 3-coloring. (See Proposition 1.) (Color figure online)

It is well known that a graph is 2-colorable if and only if it is bipartite. But we cannot expect such a nice characterization of 3-colorable graphs, at least not one in the algorithmic sense, because it is NP-hard to decide whether a graph is 3-colorable [3, 6]. The situation is similar for the local coloring problem. By definition, a graph admitting a local 2-coloring cannot have any path on three vertices. In other words, its maximum degree is at most one. It is easy to verify that this necessary condition is also sufficient. Very recently, Li et al. [4] proved that for any fixed $k = 4$ or $k = 2t - 1$ ($t \geq 3$), the local k -coloring problem is NP-complete, and then, Shitov [8] completed the hardness result by showing that it is NP-complete to decide whether a graph admits a local k -coloring for any $k \geq 3$.

A trivial fact on coloring is that if we add a new vertex to a graph and make the new vertex adjacent to all the old vertices, then the chromatic number increases by one. This is not true for local coloring: Such an operation may increase local chromatic number by one or two. For example, the complete graph on one, two, and three vertices have local chromatic number one, two, and four, respectively. Indeed, it is easy to verify that the local chromatic number of a complete graph on n vertices is $\lfloor \frac{3n-1}{2} \rfloor$ [1].

Theorem 1. *If we add two new vertices to a graph and make them adjacent to all the old vertices, but they are not adjacent themselves, then the local chromatic number increases by precisely two.*

This result subsumes the main result of Chartrand et al. [1], which gives a formula on the chromatic number of complete multipartite graphs.

Another consequence of Theorem 1 is a linear reduction from SAT to the local k -coloring problem for any integer $k \geq 3$, which implies a better lower bound for the problem. (To see that the reduction of [8] is not linear, note that it needs to introduce $\Theta(k)$ vertices for each clause—see step 2 of Theorem 12 of [8].) The Exponential Time Hypothesis (ETH) is the standard working hypothesis of fine-grained complexity, which aims to understand the exact time complexity of problems and to prove lower bounds. It postulates that the satisfiability problem with at most three variables per clause (3SAT) cannot be solved in $2^{o(p+q)}$ time, where p and q denote the number of clauses and variables respectively in the Boolean formula [5].

Theorem 2. *Assume ETH, there is no algorithm running in $2^{o(|V(G)|)}n^{O(1)}$ time that solves the local coloring problem.*

We also present a new reduction for the NP-hardness of the local coloring problem. Although the fact is already proved by Shitov [8], we believe our new proof has its own structural interest.

2 Main Results

All graphs discussed in this paper are undirected and simple. The vertex set and edge set of a graph G are denoted by $V(G)$ and $E(G)$ respectively. By $G[S]$ we denote the subgraph induced by vertex set $S \subseteq V(G)$, whose vertex set is S , and whose edge set consists of all edges with both endpoints in S . Given two disjoint graphs G_1 and G_2 , the *join* of G_1 and G_2 , denoted by $G_1 \vee G_2$, is obtained by adding edges connecting every vertex of G_1 to every vertex of G_2 . For a positive integer t , let I_t denote the graph on t vertices and with no edges.

A *coloring* of a graph G is a function $c : V(G) \rightarrow \mathbb{N}$ such that $c(u) \neq c(v)$ for all edges $uv \in E(G)$. A *local coloring* is a function $c : V(G) \rightarrow \mathbb{N}$ such that

$$\max_{v \in S} c(v) - \min_{v \in S} c(v) \geq |E(G[S])| \quad \text{for all } S \subseteq V(G), |S| \leq 3. \quad (\star)$$

Note that a local coloring is also a valid coloring: The requirement of coloring corresponds to $|S| = 2$ in (\star) . The *local chromatic number* of a graph G is defined as

$$\chi_\ell(G) = \min_{c: \text{local colorings of } G} \max_{v \in V(G)} c(v).$$

Though the definition bears striking resemblance with the chromatic number of graphs, some care needs to be taken. In coloring we want to minimize the number

of colors *actually used*, while for the local coloring we are concerned about the largest number. (Therefore, it does not make much sense to talk about how many colors a local coloring uses.) On the other hand, if the local chromatic number of graph G is ℓ and if c is a local ℓ -coloring of G , then there must be vertices receiving ℓ and 1. (Otherwise decreasing every color by 1 gives a smaller coloring.) Another way to see this is the following fact. Given any local ℓ -coloring c for G , the assignment $c'(v) = \ell + 1 - c(v)$ for all $v \in V(G)$ is also a local ℓ -coloring of G . We say that c and c' are *symmetric*.

If there exists a vertex of degree two or more, we can find a three-vertex path or a triangle, which forces us to use at least three colors. Therefore, $\chi_\ell(G) = 2$ if and only if the maximum degree of vertices in G is one (i.e., every component of G is either an isolated vertex or a single edge). The following is the technical version of Theorem 1.

Theorem 3. *For any graph G and any integer $t \geq 2$, we have $\chi_\ell(G \vee I_t) = \chi_\ell(G) + 2$.*

Proof (Proof of Theorem 1). Any local k -coloring c for G can be extended to a local $(k+2)$ -coloring c' for $G \vee I_t$ by setting $c'(u) = k+2$ for all $u \in I_t$. Therefore, $\chi_\ell(G \vee I_t) \leq \chi_\ell(G) + 2$.

Now suppose that c' is a local k' -coloring for $G \vee I_t$. Let u_1, u_2 be any two vertices in I_t and let G' denote the subgraph of $G \vee I_t$ induced by $V(G) \cup \{u_1, u_2\}$. Clearly, c' , when restricted to this vertex set, is also a local k' -coloring of G' . Let $p_1 = c'(u_1)$ and $p_2 = c'(u_2)$ and assume without loss of generality $p_1 \leq p_2$. Note that $c'(v) \notin \{p_1, p_2\}$ for all $v \in V(G)$. We give a local $(k' - 2)$ -coloring c of G as follows:

$$c(v) = \begin{cases} c'(v) - 2 & \text{if } c'(v) > p_2, \\ c'(v) - 1 & \text{if } p_1 < c'(v) < p_2, \\ c'(v) & \text{if } c'(v) < p_1. \end{cases}$$

We now argue that c satisfies (\star) . Let $X \subseteq V(G)$ and $|X| \leq 3$. Denote by $V_1 = \{v \in V(G) : c'(v) < p_1\}$, $V_2 = \{v \in V(G) : p_1 < c'(v) < p_2\}$, and $V_3 = \{v \in V(G) : c'(v) > p_2\}$, respectively. They form a partition of $V(G)$, and note that some of the three parts may be empty. If $X \subseteq V_i$ for some $i \in \{1, 2, 3\}$, then $\max_{v \in X} c(v) - \min_{v \in X} c(v) = \max_{v \in X} c'(v) - \min_{v \in X} c'(v)$. In the rest, we may assume otherwise. In other words, at least one of the following is true:

$$\min_{v \in X} c'(v) < p_1 < \max_{v \in X} c'(v), \quad \min_{v \in X} c'(v) < p_2 < \max_{v \in X} c'(v). \tag{1}$$

Note that if $p_1 = p_2$, then $|c'(v) - p_1| \geq 2$ for all $v \in v(G)$. Therefore, through some simple calculations, we always have

$$c'(y) - c'(x) \geq \begin{cases} 3 & \text{if } x \in V_1, y \in V_3, \\ 2 & \text{if } x \in V_1, y \in V_2, \\ 2 & \text{if } x \in V_2, y \in V_3. \end{cases}$$

As a result, $c(x) = c(y)$ if and only if $c'(x) = c'(y)$, for any two vertices $x, y \in V(G)$. Hence, (\star) is always satisfied if $G[X]$ has only one edge, and it suffices to consider set X with at least two edges in $G[X]$. Such a set necessarily has three vertices; let them be v_1, v_2, v_3 . We may assume without loss of generality $c'(v_1) \leq c'(v_2) \leq c'(v_3)$; note that $c(v_1) \leq c(v_2) \leq c(v_3)$ as well.

If $c(v_1) = c(v_2)$, then $v_1v_2 \notin E(G)$, and $G[X]$ is a path, namely $v_1v_3v_2$. Since both $v_1u_1v_2$ and $v_1u_2v_2$ are paths in G' , it can be inferred that $c'(v_1) \leq p_1 - 2$ when $v_1, v_2 \in V_1$, and $c'(v_1) \leq p_2 - 2$ when $v_1, v_2 \in V_2$. In either case, $c(v_3) \geq c(v_1) + 2$. It is similar when $c(v_2) = c(v_3)$.

In the rest we assume $c(v_1) < c(v_2) < c(v_3)$; now that $c(v_3) - c(v_1) \geq 2$, it suffices to consider that $G[X]$ is a triangle.

In the first case, $v_1 \in V_1$ and $v_3 \in V_3$. If $v_2 \in V_2$, then $c'(v_3) - c'(v_1) = c'(v_3) - c'(v_2) + c'(v_2) - c'(v_1) \geq 3 + 3 = 6$ because both $v_1u_1v_2$ and $v_2u_2v_3$ are triangles of G' . If $v_2 \in V_1$, then $c'(v_3) - c'(v_1) = c'(v_3) - c'(u_1) + c'(u_1) - c'(v_1) \geq 2 + 3 = 5$ because $v_1v_2u_1$ is a triangle of G' . It is similar when $v_2 \in V_3$. Thus, we can always conclude $c(v_3) - c(v_1) = c'(v_3) - c'(v_1) - 2 \geq 3$.

In the final case, $X \subseteq V_1 \cup V_2$ or $X \subseteq V_2 \cup V_3$. Then $V_2 \neq \emptyset$ and $p_1 < p_2$. Let u be the one in u_1, u_2 such that $c'(v_1) < c'(u) < c'(v_3)$. Then $c'(v_3) - c'(v_1) = c'(v_3) - c'(u) + c'(u) - c'(v_1) \geq 1 + 3 = 4$ because v_1v_2u and v_2v_3u are triangles of G' . Therefore, $c(v_3) - c(v_1) = c'(v_3) - c'(v_1) - 1 \geq 3$.

This implies $\chi_\ell(G) \leq \chi_\ell(G \vee I_t) - 2$, or equivalently $\chi_\ell(G \vee I_t) \geq \chi_\ell(G) + 2$. Therefore, they have to be equal, and the proof is now complete. \square

As we have mentioned, $\chi_\ell(G \vee I_t) - \chi_\ell(G)$ may be one or two when $t = 1$. Using Theorem 1 inductively, one can easily conclude the following corollary.

Corollary 1. *Let H be a complete p -partite graph where each part consists of at least 2 vertices. For each graph G , it holds $\chi_\ell(G \vee H) = \chi_\ell(G) + 2p$.*

Together with the fact that $\chi_\ell(K_n) = \lfloor \frac{3n-1}{2} \rfloor$, where K_n is the complete graph on n vertices, Corollary 1 gives a simpler proof for the main theorem (Theorem 3.1) of Chartrand et al. [1].

Proof (Proof of Theorem 2). Consider first $k \geq 3$ being odd. We may assume without loss of generality $3 \leq k < 2n - 1$. Given a NAE-3SAT [7] formula F , we can use the reduction of Shitov [8] to build a graph G such that G is local 3-colorable if and only if F is satisfiable. Let $p = (k - 3)/2$, and let H be a complete p -partite graph with 2 vertices in each part. Then $G \vee H$ is local k -colorable if and only if F is satisfiable. Note that the reduction of [8] is a linear reduction, and H has $O(|V(G)|)$ vertices. Therefore, the graph $G \vee H$ has a linear number of vertices. If it can be solved in $2^{O(|V(G \vee H)|)} n^{O(1)}$ time, then we can use it to solve NAE-3SAT in $2^{O(a+b)} a^{O(1)}$ time, where a and b denote the number of variables and clauses in F . However, this contradicts ETH because of a trivial linear reduction from 3SAT to NAE-3SAT.

The reduction for even $k \geq 4$ is similar: We may start from using the linear reduction from NAE-3SAT to local 4-coloring by Shitov [8] and proceed similarly as above with $p = (k - 4)/2$. Now the proof is finished. \square

The following Lemma shows a property of an optimal local coloring for a complete p -partite graph, in which each part has size at least two.¹

Lemma 1 ([1]). *Let G be a complete p -partite graph where each part consists of at least 2 vertices. Then $\chi_\ell(G) = 2p - 1$. Moreover, any local $(2p - 1)$ -coloring of G must assign the same color to all vertices in the same part.*

One may ask whether a similar statement as the second part of Lemma 1 holds for Corollary 1. As shown by the simplest example, G being a single vertex and $p = 1$, it does not. (See Fig. 2a for another example.) To have a similar conclusion, we need to have at least three vertices in each part. Let $Z(p, q)$ denote the graph obtained from a complete p -partite graph, where each part has q vertices, by adding an edge between two arbitrary vertices in the p th part. We call the last part the *anchor* of $Z(p, q)$, and the endpoints of the additional edge its *balancers*. See Fig. 2 for $Z(2, 2)$ and $Z(3, 3)$.

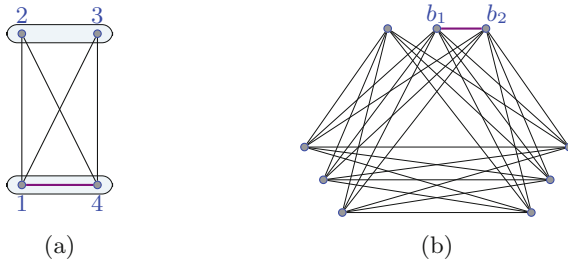


Fig. 2. (a) Graph $Z(2, 2)$, with a local 4-coloring in which vertices in the first part receive different colors. (b) Graph $Z(3, 3)$, with balancers b_1 and b_2 , has local chromatic number 6. If we remove the edge between b_1 and b_2 , we obtain $K_{3,3,3}$, whose local chromatic number is 5. (Color figure online)

Lemma 2. *Let $p, q \geq 2$ be two integers. Then $\chi_\ell(Z(p, q)) = 2p$. Moreover, if $q \geq 3$, then in any local $2p$ -coloring of $Z(p, q)$,*

- (1) *all vertices in the last part receive colors $2i - 1$ or $2i$ for some $1 \leq i \leq p$; and*
- (2) *all vertices of each of the other parts receive the same color.*

Proof. We show first $\chi_\ell(Z(p, q)) > 2p - 1$. Denote by x_1, x_2 the balancers of $Z(p, q)$. Note that removing the edge between them leaves a complete p -partite graph, of which each part has size q ; let $G' = Z(p, q) - x_1x_2$. Suppose that c is a local $(2p - 1)$ -coloring of $Z(p, q)$. Since local colorings are monotone, c is also a local $(2p - 1)$ -coloring of G' as well. By Lemma 1, all vertices in the anchor part of $Z(p, q)$, including both balancers, receive the same color in c , i.e.,

¹ Chartrand et al. [1] stated the first part as Theorem 2.4, and mentioned the second part immediately after the proof of this theorem.

$c(x_1) = c(x_2)$, which is impossible. To furnish a local $2p$ -coloring for $Z(p, q)$, it suffices to take a local $(2p - 3)$ -coloring for other vertices than the anchor, as specified in Lemma 1, and then assign $2p$ to x_1 , and $2p - 1$ to all other vertices in the anchor (including x_2).

We use induction to prove the second assertion of the lemma. It is easy to verify that in the base case $p = 2$, we have essentially only two local 4-colorings. Either assign 3 and 4 to vertices in the anchor and 1 to all other vertices, or the symmetric way. The conditions hold true for both of them. Now suppose that they hold true for p , we consider $G = Z(p + 1, q)$. By the first assertion, there must be a local $(2p + 2)$ -coloring c for G . Note that $c(x) = 2p + 2$ for at least one vertex x .

If x is not from the anchor, then there must be another vertex $x' \neq x$ such that $c(x') = 2p + 2$; otherwise $G - x$, which contains $Z(p + 1, q - 1)$ as a subgraph, can be locally colored by $2p + 1$ colors, contradicting the first assertion. Since there cannot be an edge between x and x' , they have to be from the same part; let X denote all vertices from this part. As a result, no vertex from another part, which is adjacent to both x and x' , can be colored $2p + 1$ or $2p + 2$. In other words, the coloring c , restricted to $V(G) \setminus X$, is a local $2p$ -coloring for the subgraph $G - X$, which is $Z(p, q)$. By inductive hypothesis, this coloring satisfies both conditions. Thus, it remains to show that all vertices in X receives $2p + 2$. Either (i) the two balancers receive $2p - 1$ and $2p$ respectively; or (ii) vertices in a whole part (different from X and the anchor) receive $2p$. In either case, vertices of X have to be colored $2p + 2$.

Now that x is from the anchor, all vertices that receive $2p + 2$ are also from this part. We argue first that one of the balancers receives color $2p + 2$. Otherwise, $G - \{v : c(v) = 2p + 2\}$ contains both balances and at least two vertices from each other part, and hence contains $Z(p + 1, 2)$ as a subgraph, but it admits a local $(2p + 1)$ -coloring, contradicting the first assertion. In the rest we may assume that x is a balancer; note that $c(y) \neq 2p + 2$, where y is the other balancer.

Case 1, two or more vertices in the anchor receive a color different from $2p + 2$. Note that $G - \{v : c(v) = 2p + 2\}$ contains a complete $(p + 1)$ -partite graph of which each part contains at least two vertices, and restricting c to it gives a local $(2p + 1)$ -coloring. By Lemma 1, vertices in each part have to receive the same color. In particular, $2p + 1$ has to be assigned to the other vertices in the anchor, because x is adjacent to all vertices not in this part.

Case 2, all vertices but y in the anchor receive $2p + 2$. It remains to show that $c(y) = 2p + 1$. Suppose for contradiction that $c(y) \leq 2p$. We can find a subgraph $Z(p, q)$ as follows. First, delete all other vertices in the anchor. Second, choose an arbitrary part that is not the anchor, delete one vertex from this part, and let X denote the rest vertices in this part. Third, delete all but one edge between y and X . The remaining graph is a $Z(p, q)$ with $X \cup \{y\}$ as the anchor. The coloring c restricted to it is a local $2p$ -coloring for this subgraph, and it satisfies condition (1) by inductive hypothesis; recall that $2p + 1$ is forbidden to assign to vertices not in the anchor. But then $|c(y) - c(v)| \leq 1$ for all $v \in X$, which is

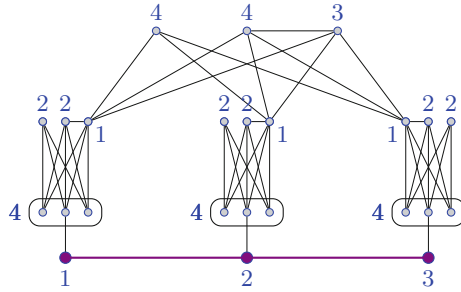


Fig. 3. Illustration for the reduction used in Lemma 3. The graph G , with three vertices and two edges, is shown at the bottom. The n copies of $Z(2, 3)$ are in the middle, and vertices x, y , and z at the top. The whole graph with all vertices is G' . The numbers, which indicate the colors, are both a local 4-coloring for G' and a valid local 3-coloring for G . (Color figure online)

impossible. Therefore, we can conclude $c(y) = 2p + 1$, and condition (2) holds following from Lemma 1, applied to $G - \{v : c(v) \geq 2p + 1\}$. This concludes the proof. \square

Lemma 2 allows us, among others, to derive a linear reduction from the local 3-coloring problem to the local 4-coloring problem (Fig. 3).

Lemma 3. *There is a linear reduction from the local 3-coloring problem to the local 4-coloring problem.*

3 Local 3-Coloring

We give an alternative reduction for the local 3-coloring problem, from the positive one-in-three SAT problem. The input of the *positive one-in-three SAT* problem consists of a set X of variables, and a CNF formula, of which each clause contains exactly three positive literals. The task of the problem is to find an assignment such that each clause has exactly one true literal, and two false literals. The NP-hardness of positive one-in-three SAT follows from the general result of Schaefer [7].

We will need the following simple fact on local 3-coloring of graphs.

Proposition 1. *Let c be a local 3-coloring of a graph G and $v \in V(G)$. If $c(v) = 2$, then $d(v) \leq 2$.*

Proof. Suppose for contradiction that $c(v) = 2$ but $d(v) \geq 3$. Then $c(u) \in \{1, 3\}$ for all $u \in N(v)$, and we can find two vertices from $N(v)$ that both receive 1 or both receive 3. But these two vertices induce a path or cycle with v , a contradiction to (\star) . \square

We introduce two gadgets, $H(x, y, z)$ and $F(a)$; see Fig. 4. The first gadget ensures that the three degree-four vertices cannot receive the same color in a local 3-coloring.

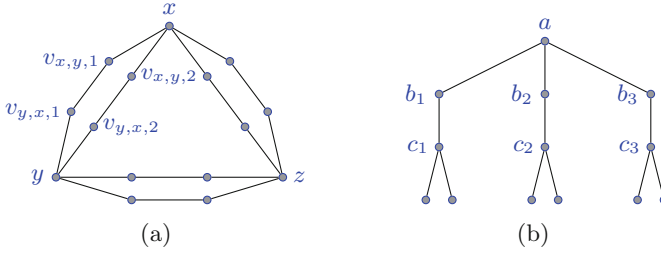


Fig. 4. Gadgets (a) $H(x, y, z)$ and (b) $F(a)$. (Color figure online)

Proposition 2. *The gadget $H(x, y, z)$ is local 3-colorable, and any local 3-coloring c of $H(x, y, z)$ satisfies $\{c(x), c(y), c(z)\} = \{1, 3\}$.*

Proof. Since Proposition 1, either of x, y and z can be colored by only 1 or 3. Denote by $v_{a,b,1}$ and $v_{a,b,2}$ the two neighbors of a on the paths from a to b avoiding c , for distinct $a, b, c \in \{x, y, z\}$, where, $v_{a,b,i}v_{b,a,i}$ is an edge of $H(x, y, z)$ for $i = 1, 2$. For contrary, $c(x), c(y)$ and $c(z)$ are the same. Assume without loss of generality, let $c(x) = c(y) = c(z) = 1$. Then one of $v_{x,y,1}$ and $v_{x,y,2}$ is colored by 2: Otherwise, they are both colored by 3, and then the two neighbors $v_{y,x,1}$ and $v_{y,x,2}$ of y should be colored by 2, whereas $\{v_{y,x,1}, v_{y,x,2}, y\}$ induces a P_3 contradicting (\star) . For the same reason, one of $v_{x,z,1}$ and $v_{x,z,2}$ will be colored by 2. However, two neighbors of x are colored by 2, a contrary.

Now it is sufficient to give a local 3-colorable c for $H(x, y, z)$. We first color x, y, z by colors 1, 3, 3 respectively, and then color $v_{x,y,1}, v_{x,y,2}, v_{x,z,1}, v_{x,z,2}$ by color 3, and $v_{y,x,1}, v_{y,x,2}, v_{z,x,1}, v_{z,x,2}, v_{y,z,2}, v_{z,y,1}$ by color 1. In the end, we color $v_{y,z,1}$ and $v_{z,y,2}$ by color 2. It is easy to check that every edge of $H(x, y, z)$ is properly colored. Since there is no triangle in $H(x, y, z)$, we have to consider the coloring for P_3 's in it. If all vertices in a P_3 receive only colors 1 and 3, then it trivially satisfies (\star) . Now some vertex in a P_3 is colored by 2. As the coloring above, the two vertices, which receive color 2, have distance at least 3 in $H(x, y, z)$. Thus, no P_3 can contain both of them. Therefore, by a quick inspection on the P_3 's receiving color 2, the other two vertices of any such a P_3 are colored by 1 and 3. Now we have that $H(x, y, z)$ is local 3-colorable. \square

The second gadget, being bipartite, is trivially local 3-colorable. Here we are interested in a special local 3-coloring.

Proposition 3. *There is a local 3-coloring for $F(a)$ such that (1) for each $i = 1, 2, 3$, the two degree-one neighbors of c_i receive the same color; and (2) the six degree-one vertices receive two different colors. Moreover, in such a local 3-coloring, precisely two of the six vertices receive the same color as vertex a .*

Proof. To prove the statement, we only have to show such a local 3-coloring c for $F(a)$. By Proposition 1, either of $c(a)$ and $c(c_i)$ is in $\{1, 3\}$ for $i = 1, 2, 3$. Let D_i be the set of the two degree-one neighbors of c_i . Then for $p \in \{1, 3\}$, we have

$$c(v) = \begin{cases} p & \text{if } v \in \{a, c_2, c_3\} \cup D_1, \\ 4 - p & \text{if } v \in \{b_2, b_3, c_1\} \cup D_2 \cup D_3, \\ 2 & \text{otherwise (i.e., } v = b_1). \end{cases}$$

Through a quick check on $F(a)$, we know that every edge in $F(a)$ is properly colored. Since no triangle contains in $F(a)$, considering the coloring of P_3 's in $F(a)$ is sufficient. If a P_3 receives only colors 1 and 3, then it trivially satisfies (\star) . Hence, let X be a P_3 containing a vertex colored by 2. By the coloring c , X contains b_1 . If b_1 is degree-one in X , then X contains an edge colored by p and $4 - p$; otherwise, the two neighbors of b_1 are colored by p and $4 - p$. In both case, X satisfies (\star) . Thus, c is a local coloring.

For (1), it is trivially satisfied. For (2), the colors for D_1, D_2 and D_3 are $p, 4 - p, 4 - p$ respectively, and only the two vertices in D_1 receive the color of a . Now, the proof is completed. \square

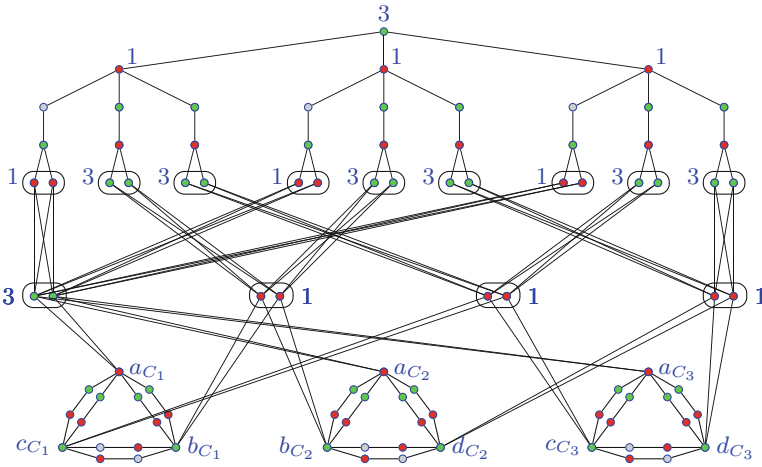


Fig. 5. A reduction from (X, \mathcal{C}) to $(G, 3)$, where $X = \{a, b, c, d\}$, and $\mathcal{C} = \{C_1 = (a, b, c), C_2 = (a, b, d), C_3 = (a, c, d)\}$. The whole graph is G , for which a local 3-coloring is indicated by colors, where red, green and grey indicate colors 1, 3 and 2 respectively. (Color figure online)

Let (X, \mathcal{C}) be an instance of the positive one-in-three SAT problem. An instance G of the local 3-coloring problem is created as follows. (1) For each clause $C = (x, y, z) \in \mathcal{C}$, we make gadgets $H_C = H(x_C, y_C, z_C)$ and $F(v_C)$. (2) For each variable $x \in X$, we make a $J_x = I_2$. (3) Let V_x denote the vertex set containing all x_C in G , for $C \in \mathcal{C}$ and $x \in X$. We join J_x and V_x , i.e., connect every vertex in J_x to every vertex in V_x . (4) For every $C = (x, y, z) \in \mathcal{C}$, let D_1, D_2 and D_3 denote a partition of degree-one vertices in $F(v_C)$, where for $i = 1, 2, 3$, D_i contains two degree-one vertices sharing a common neighbor, and

D_i is called a *bottom* of $F(v_C)$. We join D_1 and J_x , D_2 and J_y , and D_3 and J_z . (5) In the end, we make a new vertex g , and connect g to every v_C for $C \in \mathcal{C}$. See Fig. 5 for example.

Let $|X| = n$ and $|\mathcal{C}| = m$. Since $|V(F(a))| = 13$ and $|V(H(a, b, c))| = 9$, we have $|V(G)| = 1 + (13 + 9) * m + 2n = 21m + 2n + 1$. Since $|E(F(a))| = 12$ and $|E(H(a, b, c))| = 18$, we have $|E(G)| = 3 + (12 + 18) * m + 3 * 2 * 3m = 48m + 3$. Therefore, the reduction can be done in linear time.

Lemma 4. *The instance (X, \mathcal{C}) is YES if and only if G is local 3-colorable.*

4 Concluding Remarks

Let $\chi(G)$ and $\omega(G)$ denote, respectively, the chromatic number and the size of maximum cliques of a graph G . A trivial fact is

$$\chi(G) \geq \omega(G) \tag{2}$$

for all graphs G . Actually, the study of graphs on which they are equal led to the fruitful perfect graph theory. Recall that a graph G is *perfect* if $\chi(H) = \omega(H)$ for all induced subgraphs H of G . A natural question can be raised on the local chromatic number of perfect graphs. One can easily generalize (2) to

$$\chi_\ell(G) \geq \chi_\ell(K_{\omega(G)}). \tag{3}$$

However, they are not always equal on perfect graphs, e.g., consider the graph obtained by deleting one edge from K_5 .

On the algorithmic aspect, the chromatic number of perfect graphs can be computed in polynomial time. But we conjecture that the local chromatic number of perfect graphs is NP-hard. Actually, the computation of local chromatic number of split graphs, one of the simplest subclass of perfect graphs, is already nontrivial. Recall that a graph is a split graph if its vertices can be (not necessarily uniquely) partitioned into a clique and an independent set. The following is easy to verify.

Proposition 4. *Let G be a split graph. Then $\chi_\ell(G) = \chi_\ell(K_{\omega(G)})$ or $\chi_\ell(K_{\omega(G)}) + 1$.*

But to decide which one is the case is equivalent to the following variation of the hitting set problem: Notice that if a problem is NP-hard on split graphs, then it is NP-hard on perfect graphs.

Given a family S_1, \dots, S_p of subsets of U , is there a set of disjoint pairs in U such that S_i contains at least one of the pairs for all $i = 1, \dots, p$?

References

1. Chartrand, G., Saba, F., Salehi, E., Zhang, P.: Local colorings of graphs. *Utilitas Math.* **67**, 107–120 (2005)
2. Chartrand, G., Salehi, E., Zhang, P.: On local colorings of graphs. *Congr. Numer.* **163**, 207–221 (2003)
3. Garey, M.R., Johnson, D.S., Sockeye, L.J.: Some simplified NP-complete graph problems. *Theor. Comput. Sci.* **1**, 237–267 (1976). [https://doi.org/10.1016/0304-3975\(76\)90059-1](https://doi.org/10.1016/0304-3975(76)90059-1)
4. Li, Z., Zhu, E., Shao, Z., Jin, X.: NP-completeness of local colorings of graphs. *Inf. Process. Lett.* **130**, 25–29 (2018). <https://doi.org/10.1016/j.ipl.2017.09.013>
5. Impagliazzo, R., Paturi, R.: On the complexity of k -SAT. *J. Comput. Syst. Sci.* **62**(2), 367–375 (2001). <https://doi.org/10.1006/jcss.2000.1727>
6. Lovász, L.: Coverings and colorings of hypergraphs. In: *Proceedings of the Fourth Southeastern Conference on Combinatorics, Graph Theory and Computing*, pp. 3–12 (1973)
7. Schaefer, T.J.: The complexity of satisfiability problems. In: *Proceedings of the 10th Annual ACM Symposium on Theory of Computing*, San Diego, California, pp. 216–226 (1978). <https://doi.org/10.1145/800133.804350>
8. Shitov, Y.: On the complexity of graph coloring with additional local conditions. *Inf. Process. Lett.* **135**, 92–94 (2018). <https://doi.org/10.1016/j.ipl.2018.03.009>



Secure Computation of Any Boolean Function Based on Any Deck of Cards

Kazumasa Shinagawa^{1,2(✉)} and Takaaki Mizuki³

¹ Tokyo Institute of Technology, Meguro, Japan
shinagawakazumasa@gmail.com



² Institute of Advanced Industrial Science and Technology (AIST), Kōtō, Japan

³ Tohoku University, Sendai, Japan




Abstract. It is established that secure computation can be achieved by using a deck of physical cards. Almost all existing card-based protocols are based on a *specific* deck of cards. In this study, we design card-based protocols that are executable using *any* deck of cards (e.g., playing cards, UNO, and trading cards). Specifically, we construct a card-based protocol for any Boolean function based on any deck of cards. As corollaries of our result, a standard deck of playing cards (having 52 cards) enables secure computation of any 22-variable Boolean function, and UNO (having 112 cards) enables secure computation of any 53-variable Boolean function.

Keywords: Secure computation · Card-based protocols · Playing cards

1 Introduction

Secure computation enables parties having secret inputs to compute a joint function of their inputs without revealing information about the inputs that is not trivially revealed by knowing the output. It is established that secure computation can be achieved by using a deck of physical cards; this is known as card-based cryptography (e.g., [1, 2, 5]). Card-based protocols enable participants, including those unfamiliar with mathematics, to be convinced about the correctness and security of their computations. In this study, we design card-based protocols based on *general* decks of cards; almost all the existing protocols are based on a *specific* deck of cards such as a two-colored deck consisting of two types of cards:  and . We refer to this two-colored deck as a *deck of binary cards*.

1.1 A Deck of Binary Cards

A *deck of binary cards* consists of a finite number of cards whose faces display either  or  and the backs display an identical symbol . All cards with an identical symbol are indistinguishable. The following encoding rule is used:

$$\spadesuit \heartsuit = 0, \quad \heartsuit \spadesuit = 1.$$

Two face-down cards $\boxed{?}\boxed{?}$ representing a bit $x \in \{0, 1\}$ is referred to as a *commitment to x* . Given a collection of input commitments to $x_1, x_2, \dots, x_n \in \{0, 1\}$, a card-based protocol for a function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ generates a commitment to the output value $f(x_1, x_2, \dots, x_n)$ as follows:

$$\underbrace{\boxed{?}\boxed{?}}_{x_1} \underbrace{\boxed{?}\boxed{?}}_{x_2} \underbrace{\boxed{?}\boxed{?}}_{x_3} \cdots \underbrace{\boxed{?}\boxed{?}}_{x_n} \rightarrow \underbrace{\boxed{?}\boxed{?}}_{f(x_1, x_2, \dots, x_n)}.$$

The first card-based protocol is the *five-card trick* proposed by den Boer [2]. It is an AND protocol using a deck of five cards $\spadesuit \spadesuit \heartsuit \heartsuit \heartsuit$; it reveals the output value directly¹ rather than generating a commitment to the output value. Mizuki and Sone [5] showed that every Boolean function can be securely computed in a finite runtime by using a deck of binary cards; they constructed a six-card AND, a four-card XOR, and a six-card COPY protocols; these are state-of-the-art finite-runtime protocols under the condition that shuffles used in a protocol are *random cut* and/or *random bisection cut*; these shuffles are known to be physically implementable [8]. Whereas they did not consider the actual number of cards, Nishida et al. [7] showed that any n -variable Boolean function can be securely computed by using a deck of $2n + 6$ binary cards.

1.2 A Deck of Playing Cards

Whereas almost all existing protocols are based on a deck of binary cards, there are certain exceptions [4, 6]. Niemi and Renvall [6] and Mizuki [4] constructed card-based protocols based on a *standard deck of playing cards*, where all the cards are distinguishable while their backs display an identical symbol $\boxed{?}$. By creating a total order on the set of 52 cards, we can assume that a standard deck of playing cards is of the following form: $\boxed{1}\boxed{2}\boxed{3}\boxed{4}\boxed{5}\boxed{6} \cdots \boxed{52}$. The encoding rule is as follows: $\boxed{i}\boxed{j} = 0, \quad \boxed{j}\boxed{i} = 1$, where i, j are integers such that $i < j$. A *commitment to x* is defined by a pair of face-down cards with the encoding rule and denoted by $[x]^{\{i, j\}}$; here, $\{i, j\}$ is called the *base* of the commitment. Following the encoding rule, Niemi and Renvall [6] showed that any Boolean function can be securely computed by a Las-Vegas protocol. Mizuki [4] showed that it can be achieved by a *finite-runtime* protocol. Specifically, Mizuki [4] constructed an eight-card AND, a four-card XOR, and a six-card COPY protocols. Note that the eight-card AND protocol requires two more cards than the binary card protocol.

1.3 This Work: Any Deck of Cards

In this study, we design card-based protocols based on various decks of cards. Specifically, our protocols are operable on all decks in a certain class of decks: *majority-free decks*. Majority-free decks satisfy the following properties:

¹ This type of protocols is called *non-committed format*. Meanwhile, the other type of protocols, where the output is not revealed, is called *committed format*. We focus on committed format protocols throughout this paper.

1. It consists of a finite number of cards.
2. The backs of all the cards in the deck have an identical symbol.
3. For each symbol on the faces, the number of cards having the symbol does not exceed half of the total number of cards in the deck.

The third condition is necessary in order to use the two-cards-per-bit encoding. It excludes the following type of decks: $\boxed{1} \boxed{2} \boxed{1} \boxed{1} \boxed{1} \boxed{1}$. In the above-mentioned deck, whereas a commitment whose base is $\{1, 2\}$ can be created, the remaining four cards are ineffective. The third condition guarantees that we can create k commitments for $2k$ cards.

We then classify the class of majority-free decks into the following three types according to the symbols on the faces:

Type-0 A deck in which all the symbols are distinct.

Type-1 A deck in which all the symbols are distinct except for a single symbol, wherein two or more cards display this symbol.

Type-2 A deck that is neither type-0 nor type-1, i.e., a deck in which there exist two or more symbols, each of which are displayed by two or more cards.

A majority-free deck of binary cards, whose number of cards is at least four, is a type-2 deck because \clubsuit and \heartsuit are two such symbols. A standard deck of playing cards is a type-0 deck because all the cards are distinct. A deck of playing cards with two jokers whose faces are identical is a canonical example of a type-1 deck because two jokers are the exception. A deck of UNO is a type-2 deck because $\boxed{1}$ with green and $\boxed{2}$ with green are two such symbols. A deck of trading cards (e.g., “Pokémon Trading Card Game”) might be any type of deck of cards.

1.4 Our Result

In this study, we construct a card-based protocol for any Boolean function based on any majority-free deck of cards. Our result is summarized by Theorem 1:

Theorem 1. *Let f be any n -variable Boolean function. Then, for any $i \in \{0, 1, 2\}$, we can securely compute f using a type- i majority-free deck of $2n + 8 - i$ cards.*

Table 1 presents a comparison between the previous work by Nishida et al. [7] and our work. It is noteworthy that our result shows that *any* type-2 deck provides a protocol for any function as efficient (in terms of the number of cards) as the deck of binary cards.

Our result also implies the following corollaries:

- The standard deck of playing cards (having 52 cards), which is a type-0 deck, enables secure computation of any 22-variable Boolean function.
- A deck of playing cards with a pair of jokers and a spare card (55 cards in total), which is a type-1 deck, enables secure computation of any 24-variable Boolean function.
- A deck of UNO (having 112 cards), which is a type-2 deck, enables secure computation of any 53-variable Boolean function.

Table 1. A comparison between our work and the existing work

	Type of deck	# of cards
Nishida et al. [7]	binary (type-2)	$2n + 6$
Nishida et al. [7] & Mizuki [4]	type-0	$2n + 8$
Ours	type-1	$2n + 7$
Ours	type-2	$2n + 6$

1.5 Related Works

Koch et al. [3] showed the effectiveness of *non-uniform* and/or *non-closed* shuffles by constructing a four-card Las-Vegas AND protocol and a five-card finite-runtime AND protocol using them. In contrast, we focus on constructing protocols using *uniform* and *closed* shuffles, specifically, a random bisection cut. This is because it can be conveniently implemented manually [8].

2 Preliminaries

In this section, we review a few existing protocols and define fundamental notations. In Sect. 2.1, we introduce a random bisection cut, which is a shuffle operation used in existing and our protocols. In Sects. 2.2 and 2.3, we introduce an XOR protocol [5] and an input-preserving AND protocol [7]. Although they are assumed to be used with a deck of binary cards, we will use them for various decks of cards in our protocols. In Sect. 2.4, we define a few notations for various decks of cards.

2.1 Random Bisection Cut

A *random bisection cut* is a shuffle operation, which can be applied to a sequence of $2k$ cards for any integer k . Given $2k$ cards, it first bisects the sequence of $2k$ cards into two sequences of k cards: A and B :

$$\begin{matrix} 1 & 2 & 3 & 4 & & 2k-1 & 2k \\ \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \cdots & \boxed{?} & \boxed{?} \end{matrix} \rightarrow \underbrace{\begin{matrix} 1 & 2 & \cdots & k \\ \boxed{?} & \boxed{?} & \cdots & \boxed{?} \end{matrix}}_A \underbrace{\begin{matrix} k+1 & k+2 & \cdots & 2k \\ \boxed{?} & \boxed{?} & \cdots & \boxed{?} \end{matrix}}_B,$$

and then switches them randomly. Each case occurs with a probability of $1/2$:

$$\underbrace{\begin{matrix} 1 & 2 & \cdots & k \\ \boxed{?} & \boxed{?} & \cdots & \boxed{?} \end{matrix}}_A \underbrace{\begin{matrix} k+1 & k+2 & \cdots & 2k \\ \boxed{?} & \boxed{?} & \cdots & \boxed{?} \end{matrix}}_B \text{ or } \underbrace{\begin{matrix} k+1 & k+2 & \cdots & 2k \\ \boxed{?} & \boxed{?} & \cdots & \boxed{?} \end{matrix}}_B \underbrace{\begin{matrix} 1 & 2 & \cdots & k \\ \boxed{?} & \boxed{?} & \cdots & \boxed{?} \end{matrix}}_A.$$

We require that nobody can learn which case will occur. Ueda et al. [8] demonstrated that a random bisection cut is implementable physically.

2.2 Existing XOR Protocol

Using a deck of binary cards, Mizuki and Sone [5] constructed a four-card XOR protocol and a six-card AND protocol. The four-card XOR protocol accepts two commitments to $x_1, x_2 \in \{0, 1\}$ as inputs and outputs a commitment to the XOR value $x_1 \oplus x_2$ as follows:

$$\underbrace{\boxed{?} \boxed{?} \boxed{?} \boxed{?}}_{x_1} \rightarrow \underbrace{\boxed{\clubsuit} \boxed{\heartsuit}}_0 \underbrace{\boxed{?} \boxed{?}}_{x_1 \oplus x_2}.$$

We use a modified version of the protocol: a six-card XOR protocol; it accepts three commitments to $x_1, x_2, x_3 \in \{0, 1\}$ as inputs and outputs two commitments to $x_1 \oplus x_2$ and $x_1 \oplus x_3$ as follows:

$$\underbrace{\boxed{?} \boxed{?} \boxed{?} \boxed{?} \boxed{?} \boxed{?}}_{x_1} \rightarrow \underbrace{\boxed{\clubsuit} \boxed{\heartsuit}}_0 \underbrace{\boxed{?} \boxed{?}}_{x_1 \oplus x_2} \underbrace{\boxed{?} \boxed{?}}_{x_1 \oplus x_3}.$$

The four-card XOR protocol is immediately obtained from the six-card XOR protocol by omitting the rightmost two cards. The construction of the XOR protocol is omitted due to the page limitation.

2.3 Existing Input-Preserving and Protocol

Using a deck of binary cards, Mizuki and Sone [5] constructed a six-card AND protocol; it accepts two commitments to $x_1, x_2 \in \{0, 1\}$ as inputs and outputs a commitment to the AND value $x_1 x_2$. Nishida et al. [7] improved it to a six-card input-preserving AND (hereafter, IP-AND) protocol; it outputs a commitment to the AND value $x_1 x_2$ together with the input commitment to x_2 as follows:

$$\underbrace{\boxed{?} \boxed{?} \boxed{?} \boxed{?} \boxed{?} \boxed{?}}_{x_1} \rightarrow \underbrace{\boxed{\clubsuit} \boxed{\heartsuit}}_0 \underbrace{\boxed{?} \boxed{?}}_{x_2} \underbrace{\boxed{?} \boxed{?}}_{x_1 x_2}.$$

The construction of the IP-AND protocol is omitted due to the page limitation.

2.4 Notations for Various Decks of Cards

Without loss of generality, we can assume that each card has a natural number on the face as follows:

$$\boxed{1} \boxed{2} \boxed{3} \boxed{4} \boxed{5} \boxed{6} \cdots \boxed{n}.$$

As in [4], a commitment to $x \in \{0, 1\}$ of base $\{i, j\}$ is denoted by

$$\underbrace{\boxed{?} \boxed{?}}_{[x]^{\{i, j\}}}.$$

When we do not consider the base of the commitment, we denote it by

$$\underbrace{\boxed{?} \boxed{?}}_x.$$

For any type-2 deck of cards, we assume that each of the numbers of $\boxed{1}$ and $\boxed{2}$ is at least 2 and call the base $\{1, 2\}$ a *special base of type-2*. Similarly, for any type-1 deck of cards, we assume that the number of $\boxed{1}$ is at least 2 and call the base containing 1 a *special base of type-1*. Unlike the type-2 case, the other card of a special base of type-1 is arbitrary. For any type- i deck, $i \in \{1, 2\}$, a commitment having a special base is denoted by

$$\underbrace{\boxed{?} \boxed{?}}_{[x]^\dagger}.$$

We note that we can create at least two commitments having a special base because type-2 decks contain two pairs of $\boxed{1} \boxed{2}$, and type-1 decks contain two $\boxed{1}$. Although the dagger \dagger is inconsequential for a type-0 deck, we use the notation $[\cdot]^\dagger$ even for a type-0 deck in order to express type-0/1/2 commitments simultaneously.

We denote a face-up card by $\boxed{*}$ when we do not care about the face-up symbol of the card. For example, a sequence of face-up cards $\boxed{1} \boxed{3} \boxed{4} \boxed{5}$ can be denoted by

$$\boxed{1} \boxed{*} \boxed{*} \boxed{*}.$$

Here, the special card $\boxed{1}$ is explicitly written.

3 Our Input-Preserving and Protocol

In this section, our IP-AND protocol is presented. The key primitive is an *opaque commitment pair* (OC pair) introduced by Mizuki [4]. In Sect. 3.1, an OC pair is introduced. In Sect. 3.2, we present a new technique for producing an OC pair. In Sect. 3.3, we present our construction.

3.1 Opaque Commitment Pair

We first explain why a straight-forward application of the existing IP-AND protocol is ineffective for a general deck of cards. Suppose that the following sequence is input to the existing IP-AND protocol:

$$\underbrace{\boxed{?} \boxed{?}}_{[x_1]^{\{1,2\}}} \underbrace{\boxed{?} \boxed{?}}_{[x_2]^{\{3,4\}}} \underbrace{\boxed{?} \boxed{?}}_{[0]^{\{5,6\}}}.$$

In the last step of the protocol [7] (Sect. 2.1: Improved AND Protocol), the commitment to x_1x_2 is turned over, and it reveals whether the base of the

commitment is $\{3, 4\}$ or $\{5, 6\}$. The former case implies that $x_1x_2 = x_2$, and the latter case implies that $x_1x_2 = 0$; these imply that $x_1 = 1$ and $x_1 = 0$, respectively. Therefore, the secret input x_1 is revealed publicly.

Mizuki [4] solved the above problem by producing an OC pair, which is a pair of commitments of two bases such that it is unknown as to which commitment is of which base. For example, we apply a random bisection cut to a pair of two commitments of bases $\{1, 2\}$ and $\{3, 4\}$ as follows:

$$\underbrace{\boxed{?}\boxed{?}}_{[x]^{\{1,2\}}} \underbrace{\boxed{?}\boxed{?}}_{[x]^{\{3,4\}}} \rightarrow \left[\boxed{?}\boxed{?} \mid \boxed{?}\boxed{?} \right] \rightarrow \underbrace{\boxed{?}\boxed{?}}_x \underbrace{\boxed{?}\boxed{?}}_x;$$

Then, the pair becomes an OC pair because it is unknown as to which commitment is of base $\{1, 2\}$. We denote the above commitment by $[x]^{\{1,2\},\{3,4\}}$. Mizuki's technical idea is to use an OC pair $[x_2]^{\{3,4\},\{5,6\}}$ and $[0]^{\{3,4\},\{5,6\}}$ rather than $[x_2]^{\{3,4\}}$ and $[0]^{\{5,6\}}$. Now, the existing IP-AND protocol is effective because the (revealed) base of the commitment to x_1x_2 is independent of the secret input x_1 .

We call a protocol for producing an OC pair of $(x, 0)$ from a commitment to x an *OC pair generation*. Mizuki's OC pair generation for type-0 decks proceeds as follows:

1. Place six cards as follows:

$$\underbrace{\boxed{?}\boxed{?}}_{[x]^{\{5,6\}}} \boxed{1} \boxed{2} \boxed{3} \boxed{4} \rightarrow \underbrace{\boxed{?}\boxed{?}}_{[x]^{\{5,6\}}} \underbrace{\boxed{?}\boxed{?}}_{[0]^{\{1,2\}}} \underbrace{\boxed{?}\boxed{?}}_{[0]^{\{3,4\}}}.$$

2. Apply a random bisection cut to the rightmost four cards; then, we have an opaque commitment pair to two 0s as follows:

$$\underbrace{\boxed{?}\boxed{?}}_{[x]^{\{5,6\}}} \underbrace{\boxed{?}\boxed{?}}_{[0]^{\{1,2\},\{3,4\}}} \underbrace{\boxed{?}\boxed{?}}_{[0]^{\{1,2\},\{3,4\}}}.$$

3. Apply the four-card XOR protocol (Sect. 2.2); then, we have an OC pair of $(x, 0)$:

$$\boxed{5} \boxed{6} \underbrace{\boxed{?}\boxed{?}}_{[x]^{\{1,2\},\{3,4\}}} \underbrace{\boxed{?}\boxed{?}}_{[0]^{\{1,2\},\{3,4\}}}.$$

For type-2 decks, an OC pair is not required when the input commitment has the special base $\{1, 2\}$ and two cards $\boxed{1} \boxed{2}$ are free. We regard the following trivial protocol as an OC pair generation for type-2 decks:

1. Place four cards as follows:

$$\underbrace{\boxed{?}\boxed{?}}_{[x]^{\{1,2\}}} \boxed{1} \boxed{2}.$$

2. Turning over the face-up cards, we have

$$\underbrace{\boxed{?} \boxed{?}}_{[x]^{\{1,2\}}} \underbrace{\boxed{?} \boxed{?}}_{[0]^{\{1,2\}}}.$$

Now, we have OC pair generations for type-0 and type-2 decks. In the next section, we present a new technique of OC pair generation for type-1 decks.

3.2 New Technique of Opaque Commitment Pair Generation

In this section, we present a new technique of OC pair generation for type-1 decks. It produces an opaque commitment pair $[x]^{\{1,2\},\{1,3\}}$ and $[0]^{\{1,2\},\{1,3\}}$ from a commitment $[x]^{\{1,*\}}$ with three free cards $\boxed{1} \boxed{2} \boxed{3}$. The key observation is that the left card of the commitment $[0]^{\{1,2\},\{1,3\}}$ is always $\boxed{1}$. It proceeds as follows:

1. Place a commitment to x with three cards as follows:

$$\underbrace{\boxed{?} \boxed{?}}_{[x]^{\{1,*\}}} \boxed{1} \boxed{2} \boxed{3}.$$

2. Turn over all the face-up cards; then, apply a random bisection cut to the rightmost two cards:

$$\underbrace{\boxed{?} \boxed{?}}_{[x]^{\{1,*\}}} \boxed{?} \left[\boxed{?} \mid \boxed{?} \right].$$

3. Apply the four-card XOR protocol (Sect. 2.2) to the first and second commitments; then, we have

$$\boxed{1} \boxed{*} \underbrace{\boxed{?} \boxed{?}}_{[x]^{\{1,2\},\{1,3\}}} \boxed{?}.$$

4. Rearrange the order of the sequence according to a permutation (1 3)(2 5 4):

$$\boxed{1} \boxed{*} \underbrace{\boxed{?} \boxed{?}}_{[x]^{\{1,2\},\{1,3\}}} \boxed{?} \rightarrow \underbrace{\boxed{?} \boxed{?}}_{[x]^{\{1,2\},\{1,3\}}} \boxed{1} \boxed{?} \boxed{*}.$$

5. Turn over $\boxed{1}$; then, we have an opaque commitment pair:

$$\underbrace{\boxed{?} \boxed{?}}_{[x]^{\{1,2\},\{1,3\}}} \underbrace{\boxed{?} \boxed{?}}_{[0]^{\{1,2\},\{1,3\}}} \boxed{*}.$$

We denote an OC pair by $[x]_{\text{OC}}$ and $[0]_{\text{OC}}$. For type-0 decks, it comprises $[x]^{\{1,2\},\{3,4\}}$ and $[0]^{\{1,2\},\{3,4\}}$. For type-1 decks, it comprises $[x]^{\{1,2\},\{1,3\}}$ and $[0]^{\{1,2\},\{1,3\}}$. For type-2 decks, it comprises $[x]^{\{1,2\}}$ and $[0]^{\{1,2\}}$.

3.3 Description of Our Input-Preserving AND protocol

In this section, we construct a new IP-AND protocol by our OC pair generation technique. It proceeds as follows:

1. Place two commitments to x_1, x_2 with free cards as follows:

$$\underbrace{\boxed{?}\boxed{?}\boxed{?}\boxed{?}}_{x_1} \underbrace{\boxed{*}\boxed{*}\boxed{*}\boxed{*}\dots\boxed{*}}_{[x_2]^\dagger \text{ free cards}},$$

where the free cards are as follows:

$$\text{type-2 : } \boxed{1}\boxed{2} \quad \text{type-1 : } \boxed{1}\boxed{*}\boxed{*} \quad \text{type-0 : } \boxed{*}\boxed{*}\boxed{*}\boxed{*}.$$

2. Apply the OC pair generation technique to the commitment to x_2 with the free cards; then, we have

$$\underbrace{\boxed{?}\boxed{?}\boxed{?}\boxed{?}}_{x_1} \underbrace{\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{*}\dots\boxed{*}}_{[x_2]_{\text{OC}} \ [0]_{\text{OC}} \ \text{free cards}},$$

where the OC pair is as follows:

- type-2 : $\{1, 2\}$
- type-1 : $\{1, 2\}, \{1, 3\}$
- type-0 : $\{1, 2\}, \{3, 4\}$

and the free cards are as follows:

$$\text{type-2 : none} \quad \text{type-1 : } \boxed{*} \quad \text{type-0 : } \boxed{*}\boxed{*}.$$

3. Apply the existing IP-AND protocol to the leftmost six cards; then, we have

$$\underbrace{\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{*}\boxed{*}}_{x_2 \ [x_1 x_2]^\dagger \ \text{free cards}},$$

where the free cards are as follows:

$$\text{type-2 : } \boxed{1}\boxed{2} \quad \text{type-1 : } \boxed{1}\boxed{*}\boxed{*} \quad \text{type-0 : } \boxed{*}\boxed{*}\boxed{*}\boxed{*}.$$

4 Our Protocol for Any Boolean Function

In this section, we construct a protocol for any n -variable Boolean function using any type- i deck of $2n+8-i$ cards. This establishes Theorem 1. In Sect. 4.1, a swap protocol, which is a subprotocol of our AND-XOR protocol, is constructed. In Sect. 4.2, the AND-XOR protocol, which is a subprotocol of our main protocol, is constructed. In Sect. 4.3, the main protocol is constructed.

4.1 Swap Protocol

It appears that a general-purpose protocol could be immediately obtained by plugging our IP-AND protocol into Nishida’s AND–XOR protocol. However, this is not true for type-2 and type-1 decks. This is because the number of duplicate cards such as $\boxed{1}$ is limited. Therefore, we have to *reuse* the duplicate cards a number of times. It is feasible to reuse them by constructing a *swap protocol*, which exchanges the bases of two commitments. It proceeds as follows:

1. Place the two commitments $[x_1]^{\{1,2\}}, [x_2]^{\{3,4\}}$ with two free cards $\boxed{5}\boxed{6}$ as follows:

$$\underbrace{\boxed{?}\boxed{?}}_{[x_1]^{\{1,2\}}} \underbrace{\boxed{?}\boxed{?}}_{[x_2]^{\{3,4\}}} \boxed{5}\boxed{6} \rightarrow \underbrace{\boxed{?}\boxed{?}}_{[x_1]^{\{1,2\}}} \underbrace{\boxed{?}\boxed{?}}_{[x_2]^{\{3,4\}}} \underbrace{\boxed{?}\boxed{?}}_{[0]^{\{5,6\}}}.$$

2. Apply the four-card XOR protocol (Sect. 2.2) to the first and third commitments; then, we have

$$\boxed{1}\boxed{2} \underbrace{\boxed{?}\boxed{?}}_{[x_2]^{\{3,4\}}} \underbrace{\boxed{?}\boxed{?}}_{[x_1]^{\{5,6\}}}.$$

3. Turn over the face-up cards, and apply the four-card XOR protocol to the second and first commitments; then, we have

$$\underbrace{\boxed{?}\boxed{?}}_{[x_2]^{\{1,2\}}} \boxed{3}\boxed{4} \underbrace{\boxed{?}\boxed{?}}_{[x_1]^{\{5,6\}}}.$$

4. Turn over the face-up cards, and apply the four-card XOR protocol to the third and second commitments; then, we have

$$\underbrace{\boxed{?}\boxed{?}}_{[x_2]^{\{1,2\}}} \underbrace{\boxed{?}\boxed{?}}_{[x_1]^{\{3,4\}}} \boxed{5}\boxed{6}.$$

4.2 AND–XOR Protocol

In this subsection, we present our *AND–XOR protocol* based on various decks of cards. Given a collection of $n + 1$ commitments to $x_1, x_2, \dots, x_n, w \in \{0, 1\}$, an AND–XOR protocol produces a commitment to $w \oplus x_1 x_2 \dots x_n$ by preserving a collection of n commitments to x_1, x_2, \dots, x_n .

Our AND–XOR protocol uses $2n + 8 - i$ cards for type- i decks as follows:

1. Place $2n + 8 - i$ cards as follows:

$$\underbrace{\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}}_{[x_1]^\dagger} \underbrace{\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}}_{[x_2]^\dagger} \underbrace{\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}}_{x_3} \dots \underbrace{\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}}_{x_n} \underbrace{\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}}_{w} \underbrace{\boxed{?}\boxed{?}}_{0} \underbrace{\boxed{?}\boxed{?}}_{0} \underbrace{\boxed{*}\dots\boxed{*}}_{\text{free cards}},$$

where the free cards are as follows:

$$\text{type-2 : no cards. type-1 : } \boxed{*} \text{ type-0 : } \boxed{*}\boxed{*}$$

2. Apply the six-card XOR protocol (Sect. 2.2) to the commitments to $x_1, 0$, and 0 ; then, we have

$$\underbrace{\boxed{?}\boxed{?}}_{x_1} \underbrace{\boxed{?}\boxed{?}}_{x_1} \underbrace{\boxed{?}\boxed{?}}_{[x_2]^\dagger} \underbrace{\boxed{?}\boxed{?}}_{x_3} \cdots \underbrace{\boxed{?}\boxed{?}}_{x_n} \underbrace{\boxed{?}\boxed{?}}_w \underbrace{\boxed{*}\cdots\boxed{*}}_{\text{free cards}}.$$

with the following free cards:

$$\text{type-2 : } \boxed{1}\boxed{2} \quad \text{type-1 : } \boxed{1}\boxed{*}\boxed{*} \quad \text{type-0 : } \boxed{*}\boxed{*}\boxed{*}\boxed{*}.$$

3. For $j = 1$ to $n - 1$, adopt the following procedure:

- (a) Apply our IP-AND protocol (Sect. 3.3) to the commitment to $x_1 \cdots x_j$ and the commitment $[x_{j+1}]^\dagger$ with free cards; then, we have

$$\underbrace{\boxed{?}\boxed{?}}_{x_1} \underbrace{\boxed{?}\boxed{?}}_{x_2} \underbrace{\boxed{?}\boxed{?}}_{x_3} \cdots \underbrace{\boxed{?}\boxed{?}}_{x_n} \underbrace{\boxed{?}\boxed{?}}_{[x_1 \cdots x_{j+1}]^\dagger} \underbrace{\boxed{?}\boxed{?}}_w \underbrace{\boxed{*}\cdots\boxed{*}}_{\text{free cards}},$$

with the following free cards:

$$\text{type-2 : } \boxed{1}\boxed{2} \quad \text{type-1 : } \boxed{1}\boxed{*}\boxed{*} \quad \text{type-0 : } \boxed{*}\boxed{*}\boxed{*}\boxed{*}.$$

If $j = n - 1$, skip Step 3-(b), and go to Step 4.

- (b) If the deck is type-0, do not take action. Otherwise, apply the swap protocol to the commitment $[x_1 \cdots x_{j+1}]^\dagger$ and the commitment to x_{j+2} .

4. Apply the four-card XOR protocol (Sect. 2.2) to the commitment $[x_1 \cdots x_n]^\dagger$ and the commitment to w ; then, we have

$$\underbrace{\boxed{?}\boxed{?}}_{x_1} \underbrace{\boxed{?}\boxed{?}}_{x_2} \underbrace{\boxed{?}\boxed{?}}_{x_3} \cdots \underbrace{\boxed{?}\boxed{?}}_{x_n} \underbrace{\boxed{?}\boxed{?}}_{w \oplus x_1 \cdots x_n} \underbrace{\boxed{*}\cdots\boxed{*}}_{\text{free cards}},$$

where the free cards are as follows:

$$\text{type-2 : } \boxed{1}\boxed{2}\boxed{1}\boxed{2} \quad \text{type-1 : } \boxed{1}\boxed{*}\boxed{1}\boxed{*}\boxed{*} \quad \text{type-0 : } \boxed{*}\boxed{*}\boxed{*}\boxed{*}\boxed{*}\boxed{*}.$$

4.3 Description of Our Protocol for Any Boolean Function

In this subsection, we prove Theorem 1 by constructing a protocol for any n -variable Boolean function f based on various decks of cards.

Similar to Nishida et al.'s construction, our construction is based on the fact that any n -variable function $f(x_1, x_2, x_3, \dots, x_n)$ can be expressed as the *Shannon expansion*:

$$\begin{aligned} f(x_1, x_2, x_3, \dots, x_n) &= \bar{x}_1 \bar{x}_2 \bar{x}_3 \cdots \bar{x}_n f(0, 0, 0, \dots, 0) \oplus x_1 \bar{x}_2 \bar{x}_3 \cdots \bar{x}_n f(1, 0, 0, \dots, 0) \\ &\oplus \bar{x}_1 x_2 \bar{x}_3 \cdots \bar{x}_n f(0, 1, 0, \dots, 0) \oplus x_1 x_2 \bar{x}_3 \cdots \bar{x}_n f(1, 1, 0, \dots, 0) \\ &\oplus \cdots \oplus x_1 x_2 x_3 \cdots x_n f(1, 1, 1, \dots, 1). \end{aligned}$$

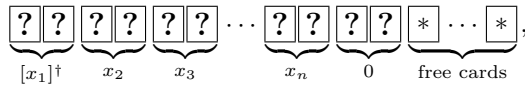
That is, the function f can be expressed as

$$f(x_1, x_2, x_3, \dots, x_n) = T_1 \oplus T_2 \oplus \dots \oplus T_\ell,$$

where each T_i is the AND value of n literals such as $\bar{x}_1 x_2 \bar{x}_3 \dots x_n$.

It proceeds as follows:

1. Place $2n + 8 - i$ cards as follows:



where the free cards are as follows:



2. Let $T_1 \oplus T_2 \oplus \dots \oplus T_\ell$ be the Shannon expansion of f . For $i = 1, 2, \dots, \ell$, add T_i to the rightmost commitment by using our AND–XOR protocol².
3. Output the rightmost commitment.

Note that the numbers of cards used in the above construction are identical to that of our AND–XOR protocols. Thus, we obtain Theorem 1.

5 Conclusion

In this study, we showed that any n -variable Boolean function can be securely computed by using a type- i majority-free deck of $2n + 8 - i$ cards for $i \in \{0, 1, 2\}$. An important open problem is to determine whether $2n + 8 - i$ cards are necessary or not. Another noteworthy open problem is to show a similar result without majority-freeness.

Acknowledgments. This work was supported in part by JSPS KAKENHI Grant Numbers 17J01169 and 17K00001.

References

1. Crépeau, C., Kilian, J.: Discreet solitary games. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 319–330. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48329-2_27
2. Boer, B.: More efficient match-making and satisfiability *the five card trick*. In: Quisquater, J.-J., Vandewalle, J. (eds.) EUROCRYPT 1989. LNCS, vol. 434, pp. 208–217. Springer, Heidelberg (1990). https://doi.org/10.1007/3-540-46885-4_23
3. Koch, A., Walzer, S., Härtel, K.: Card-based cryptographic protocols using a minimal number of cards. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015. LNCS, vol. 9452, pp. 783–807. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48797-6_32

² Prior to executing the AND–XOR protocol, for each complementary literal \bar{x}_j in T_i , we negate the corresponding commitment by swapping the two cards.

4. Mizuki, T.: Efficient and secure multiparty computations using a standard deck of playing cards. In: Foresti, S., Persiano, G. (eds.) CANS 2016. LNCS, vol. 10052, pp. 484–499. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-48965-0_29
5. Mizuki, T., Sone, H.: Six-card secure AND and four-card secure XOR. In: Deng, X., Hopcroft, J.E., Xue, J. (eds.) FAW 2009. LNCS, vol. 5598, pp. 358–369. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02270-8_36
6. Niemi, V., Renvall, A.: Solitaire zero-knowledge. *Fundam. Inform.* **38**(1–2), 181–188 (1999)
7. Nishida, T., Hayashi, Y., Mizuki, T., Sone, H.: Card-based protocols for any boolean function. In: Jain, R., Jain, S., Stephan, F. (eds.) TAMC 2015. LNCS, vol. 9076, pp. 110–121. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-17142-5_11
8. Ueda, I., Nishimura, A., Hayashi, Y., Mizuki, T., Sone, H.: How to implement a random bisection cut. In: Martín-Vide, C., Mizuki, T., Vega-Rodríguez, M.A. (eds.) TPNC 2016. LNCS, vol. 10071, pp. 58–69. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-49001-4_5



A Novel Business Model for Electric Car Sharing

Yukun Cheng¹, Xiaotie Deng², and Mengqian Zhang³(✉)

¹ Suzhou Key Laboratory for Big Data and Information Service, School of Business, Suzhou University of Science and Technology, Suzhou 215009, China

ykcheng@amss.ac.cn

² School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China

xiaotie@pku.edu.cn

³ Department of Computer Science, Shanghai Jiao Tong University, Shanghai 200240, China

mengqian@sjtu.edu.cn

Abstract. Transportation sharing in goods (bike sharing) and services (ride sharing) has been one of the most active sectors of the rising sharing economy. Such a mobile platform, based on a business model, seeks supplies in vehicles for their demands at matched times and rental locations. While its improved efficiency makes vehicle sharing popular over the traditional taxi services, it has attracted more competitors that have created redundancies in suppliers, which creates a new challenge in social efficiency. The problem also becomes for car sharing and will be intensified by the increased competition among all operators. In this work, we present a business solution by setting up a joint venture to provide shared resources as a base for different competitors to operate on. These competitors share the cost of the joint venture and then provide their own differentiated service to their customers through their own mobile apps. We formulate the number of users in the car sharing market as a S-shape function with respect to time and discretize the time into infinite rounds. In each round, we study this business model by formulating it as a two-stage Stackelberg game and apply the backward induction to analyze the equilibria of the leader and the followers in each round. Our results on the equilibria can forecast the car sharing market reasonably and avoid the redundancies in suppliers effectively. We also conduct numerical computation to visualize our theoretical results about the influence of joint venture decisions in the end.

Keywords: Electric car sharing · Joint venture · Stackelberg game · Competition · Cooperation

This research was partially supported by the National Nature Science Foundation of China (No. 11871366, 61761146005, 61632017, 61803279).

© Springer Nature Switzerland AG 2019

Y. Chen et al. (Eds.): FAW 2019, LNCS 11458, pp. 76–87, 2019.

https://doi.org/10.1007/978-3-030-18126-0_7

1 Introduction

The rising of the vehicle sharing business, propelled by techniques of mobile communication, automatic locating and tracking identification, are changing our traveling style. Among a variety of selections, including bicycle sharing and electric car sharing, vehicle-sharing industry had a phenomenal growth rate [8]. In a short time, a large number of startup companies have entered this field, with a user growth rate of 632.1% in Year 2017 alone [12].

However, the success of the few pioneers has not been sustained as the bike-sharing gold rush because of the fierce competition to the market. Within 2018, a large number of companies failed and withdrew from the market one after another [1, 2]. Even the pioneer and the past market favorite ofo has slipped into a state of bankruptcy [4]. The market badly needs a solution to save the popular bike-sharing industry from a start-up booming to a capital-suck collapsing.

Almost at the same time, electric car sharing is booming all over the world. According to 2017 China Development Report for Time-sharing Rental of New Energy Vehicle released by Roland Berger [13], the direct demand of car sharing travel users in China is about 8.16 million times per day in 2015, and it is expected to grow to 37 million times per day in 2018. The potential market capacity is expected to reach 1.8 trillion yuan. More important, the Ministry of Transport and the Ministry of Housing and Urban-Rural Development in China jointly released “Guidance on promoting the healthy development of small and micro car rental” [6], which greatly encouraged the development of the electric car sharing industry. But compared with the bicycle sharing industry, the electric car sharing industry has a higher threshold. The operators need to invest a lot of money to produce or purchase, maintain vehicles, which makes them have a higher investment risk. Now, the operators lose 50–120 yuan for each car per day on average and most of them still rely on the government subsidies [13].

The crux of the difficulty of the car sharing industry is that most operators take both of the fixed asset investment and market operations into consideration at the same time. In addition, how to predict market demand precisely and avoid the problems caused by the surplus due to oversupply, which has happened in bicycle sharing market, is also a difficulty that each operator must face.

Facing above opportunities and challenges, we present a novel business model for the electric car sharing market by establishing a joint venture. The joint venture is shared by all operators and responsible for producing and renting the cars to operators. By releasing from the burden of purchasing, producing and maintaining the cars, the operators could pay more attention to their business for the end users and the competition with the other opponents by concentrating on the quality of service. Our idea is inspired by the establishment of a joint telecom infrastructure venture, China Tower, in 2014. This joint venture is responsible for building, maintaining and operating the telecom infrastructures. Three telecom giants in China, i.e., China Mobile, China Unicom and China Telecom, agreed to co-invest and share China Tower and rent infrastructure services from it [5].

In this paper, we aim to analyze the relationship between the joint venture and the operators for the electric car sharing market in our business model.

In addition to the operators, the joint venture is also assumed to be shared by the government. This assumption makes sense, since Chinese government greatly encourages the electric car sharing market by providing the subsidies to operators at present. In our novel model, the government can also subsidize the electric car sharing industry by another way, i.e., sharing the joint venture.

Inspired by [3], we form our business model into a game system, which contains a joint venture and the competitive operators. The joint venture is shared by the operators and leases its cars to the operators. The operators provide the service to the end users by renting cars from the joint venture. Of course, all participants in this game system have their own objectives. For each operator, she tries to maximize her profit by charging the end users. For the joint venture, its social goal is to maximize the aggregate profit of the whole market. Different from the telecom market, the user scale in the electric car sharing market changes over time, while the number of users is fixed in the telecom market [3]. Hence, we refer to the user scale in the bicycle sharing market and assume the electric car sharing market also has the similar S-shape user scale [11]. Under such a setting, we discretize the time into infinite time slots (or called rounds) with the same length. And to study the interactions between the joint venture and all operators in each round, we model the electric car sharing market with the joint venture as a two-stage Stackelberg game [9]. At the first stage, the joint venture, as a leader, decides the price of renting cars and the shares of all operators in the current round. At the second stage, the operators, as the followers, set their own service prices to end users, based on the leader's decisions of the round. By applying the backward induction, we propose the closed forms of the equilibria of the joint venture and each operator in each round. Our theoretical results reveal the important influence of the joint venture on the equilibria and provide a new idea for the development of the electric car sharing market. We also conduct several numerical studies, which show the presence of the joint venture can significantly reduce the social resource consumption.

This paper is organized as follows. We will first introduce the models and preliminaries in Sect. 2. Then we conduct the equilibrium analysis in Sect. 3. In Sect. 4, we present computational experiments to verify our results and analysis. Last section concludes our work and proposes the future work.

2 Preliminaries

2.1 System Model

In this paper, we consider a market with a set $N = \{1, \dots, n\}$ of shared car operators. They compete at the end user market for further profit. These n operators invest to set up a joint venture together, which is specialized in designing, producing and maintaining the shared electric cars. Operators rent electric cars supplied by the joint venture and then provide service to their costumers from their respective mobile apps. As the shareholders of the joint venture, these operators also share the profit from it. Therefore, on the one hand, the operators

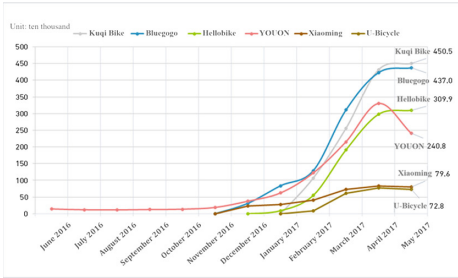
compete with each other at the end user market as before. On the other hand, they share the common benefit at the joint venture end.

2.2 User Scale Model

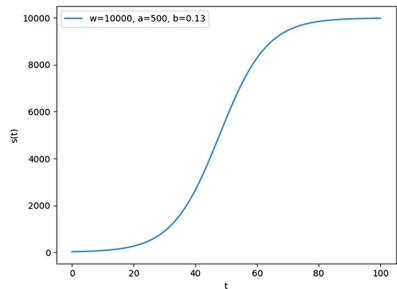
Since the electric car sharing market is an emerging market, people are not familiar with it currently and the user scale of the whole market will change over time. Therefore, we assume this change is similar to the one in the bicycle sharing market. Figure 1(a) shows how the number of users of different participants in the bicycle sharing industry changes over one year [11]. It is not hard to see that they are roughly S-shaped except for YOUON, which announced to withdraw from the competition in March 2017. Thus it's reasonable to assume that the user scale of the whole car sharing market also changes following a S shape over time. In fact, S-shaped growth can be observed in a wide variety of phenomena, such as the population growth, the spread of rumors, the cellular growth of a plant, the body's immune response and so on. As most of its applications are based on nature, it has been considered as a natural law of growth. And it's really a popular model for describing the evolution of systems (technological, economical, social and others) over time [7]. So we assume the total number of users $s(t)$ in the electric car sharing market is a S-shaped growth function on time t , and has the form as

$$s(t) = \frac{w}{1 + ae^{-bt}}, \tag{1}$$

where w is the maximum value and w, a and b are constants. Figure 1(b) shows an instance of function $s(t)$.



(a) The user scale in bike sharing market



(b) The figure of S-shape function

Fig. 1. The user scale model.

For the sake of analysis, we discretize the time into infinite timeslots with equal length Δt , denoted by $R = \{1, 2, \dots, r, \dots\}$. Each timeslot is called a round. We use superscript r to denote the value of corresponding variable in the r -th round.

2.3 Demand Model

Since the demand of each operator i is closely related to the number of the users belonging to operator i , we first propose a function to describe the number of operator i 's users in the r -th round, denoted by U_i^r . Here U_i^r contains two parts: one is the number of existing users who could flow among all operators in the previous round; the other is the number of new coming users in this round.

In our new business system, the shared cars are the same, which promises that the services from different operators are substitutes for all users. According to the classic substitute demand model in [10], the price of one player has a linear influence on the number of the users of other operators. So the first source of users' number is

$$U_i^{r-1} - mp_i^r + m \sum_{j \in N, j \neq i} \lambda_{ij} p_j^r,$$

in which p_i^r is the unit service price setting by operator i in the r -th round; m is the conversion coefficient from price to user numbers, which means raising the price by \$1 results in losing m users and λ_{ij} measures the influence of p_j^r ($j \neq i$) on U_i^r . We assume $\sum_{j \in N, j \neq i} \lambda_{ij} < 1$, which prevents all operators from raising price maliciously together.

The second source of users' number is the new coming users in the r -th round under the assumption that the user scale of the whole market follows the S-Curve. We use $\frac{wabe^{-bt}}{(1+ae^{-bt})^2}$, the first-order derivative of (1) at the middle time of the r -th round, multiplying by the length of each round Δt to approximately represent the number of new users in the r -th round. In addition, we also assume that these new coming users select the operators randomly, which promises that all operators share the new users equally in expected.

From the above, we define the user number function of operator i to be

$$U_i^r = U_i^{r-1} - mp_i^r + m \sum_{j \in N, j \neq i} \lambda_{ij} p_j^r + \frac{1}{n} \frac{wabe^{-bt}}{(1+ae^{-bt})^2} \Delta t \quad (2)$$

where $t = (r-1)\Delta t + \frac{\Delta t}{2}$, the middle time of the r -th round. Specifically, U_i^0 denotes the number of the existing users belonging to operator i before the joint venture is set up. If operator i didn't run the car-sharing business before, just set U_i^0 as 0.

Next, we model the demand of operator i in the r -th round. For the electric car sharing market, the demand is related to not only the user scale, but also how much driving they do. Although the travelling habits vary from person to person, statistical regularities behind it can be estimated by some statistical or machine learning tools. Then we define the demand function as:

$$D_i^r = U_i^r \theta. \quad (3)$$

where θ is the average request times of each user for per unit service during one round.

2.4 Profit Model

In this subsection, we will propose the profit model of the joint venture and the operators in our business model. For the joint venture, it produces the shared electric cars, rents them to all operators at a same unit price p and keep the routine maintenance. The unit cost is denoted by c , including the production cost, parking cost, maintenance cost, recharge expense and so on. Then the renting profit of the joint venture in the r -th round is given as

$$f_0^r = (p^r - c) \sum_{i \in N} D_i^r. \quad (4)$$

The profit of each operator contains two parts. After renting the shared cars from the joint venture, the operators make a profit from users by charging them for service. As an investor, the operators can also share the profit from the joint venture. We denote the share of operator i in the joint venture by α_i^r ($0 \leq \alpha_i^r \leq 1$) and $\alpha^r = \sum_{i \in N} \alpha_i^r$ (if $\alpha^r < 1$, it means there exists an outside investor, e.g., the government). The profit of operator i in the r -th round is

$$f_i^r = (p_i^r - p^r) D_i^r + \alpha_i^r f_0^r. \quad (5)$$

Finally, we define the aggregate profit of the whole market as

$$AP^r = \sum_{i \in N} f_i^r + (1 - \alpha^r) f_0^r. \quad (6)$$

For the sake of simplicity, we shall omit the superscript r in the following discuss if there is no confusion.

2.5 Game Model

As noted earlier, we discretize the time into infinite rounds. In each round, the interaction between the joint venture and the operators is modeled as a two-stage Stackelberg game. In brief, the leader (i.e., the joint venture) moves first and then the followers (i.e., all operators) moves sequentially. Specifically, we formulate the optimization problems for the leader and followers as follows.

Joint Venture's Strategies in the First Stage. In the first stage, the joint venture, as the leader, takes the current demand of the whole market as input to determine the unit renting price p and each share $\alpha_i, \forall i \in N$. In each round, the joint venture aims to maximize the aggregate profit of the entire market AP . Such a goal makes sense, because the joint venture is shared by the operators and supported by the government.

Operators' Strategies in the Second Stage. In the second stage, all operators, as the followers, take p and α_i as the inputs and decide their unit service price p_i simultaneously. Operator i aims to maximize her own profit f_i in each round, $\forall i \in N$.

In each round, these two stages together form the Stackelberg game, and the objective of this game is to find the Stackelberg equilibrium. The Stackelberg equilibrium ensures that the aggregate profit AP is maximized when all operators set their service price following the best response (i.e. the Nash equilibrium).

3 Equilibrium Analysis

In this section, we conduct a backward induction to analyze the Stackelberg game of the electric car sharing market with the joint venture in the r -th round. In Subsect. 3.1, we first analyze the competition of all operators in the second stage, under the assumption that they have observed the renting price and their shares. After analyzing their optimal strategies in the second stage, we will come back to explore the first stage of the Stackelberg game in Subsect. 3.2.

3.1 Second Stage: Operator's Profit Maximization

In this section, we analyze the competition of all operators in the second stage of the game. In order to facilitate our analysis, we first calculate the equilibrium price under the simpler setting $\lambda_{ij} = \lambda, \forall i, j \in N$, i.e. the price of each operator has the same influence on the number of each other's user, and then extend this result to general cases. In the following discuss, the superscript ne is used to denote the equilibrium value of the corresponding variable. Given the renting price p and the share $\alpha_i, i \in N$, the operator's sub-game can be written as $\max_{p_i} f_i$.

Using the previous definitions, we have

$$f_i = (p_i - p)(U_i^{r-1} - mp_i + m\lambda \sum_{j \in N, j \neq i} p_j + \frac{1}{n} \frac{wabe^{-bt}}{(1+ae^{-bt})^2} \Delta t)\theta \\ + \alpha_i(p - c)(\sum_{j \in N} U_j^{r-1} - m\lambda' \sum_{j \in N} p_j + \frac{wabe^{-bt}}{(1+ae^{-bt})^2} \Delta t)\theta$$

where $\lambda' \equiv 1 - (n - 1)\lambda$. At equilibrium, no operator could get more profit by changing her own price individually. Thus, we have $\frac{\partial f_i}{\partial p_i} = 0, \forall i \in N$, i.e.,

$$2p_i - \lambda \sum_{j \neq i} p_j = \frac{U_i^{r-1}}{m} + p - \lambda' \alpha_i(p - c) + \frac{1}{mn} \frac{wabe^{-bt}}{(1+ae^{-bt})^2} \Delta t, \forall i \in N.$$

Summing up all of n equations, the equilibrium price of operator i in the r -th round can be obtained:

$$p_i^{r,ne} = \frac{1}{2 + \lambda} (\lambda P + \frac{U_i^{r-1}}{m} + p - \lambda' \alpha_i(p - c) + \frac{1}{mn} \frac{wabe^{-bt}}{(1+ae^{-bt})^2} \Delta t). \quad (7)$$

We denote

$$P = \sum_i p_i^{r,ne} = \frac{1}{1 + \lambda'} (\frac{\sum_{i \in N} U_i^{r-1}}{m} + np - \lambda' \alpha(p - c) + \frac{1}{m} \frac{wabe^{-bt}}{(1+ae^{-bt})^2} \Delta t).$$

For general cases with arbitrary λ_{ij} , we claim that there also exists a unique equilibrium with similar format as Eq. (7). By using the same approach as above to solve $\frac{\partial f_i}{\partial p_i} = 0, \forall i \in N$, we have

$$2p_i - \sum_{j \neq i} \lambda_{ij} p_j = \frac{U_i^{r-1}}{m} + c\alpha_i(1 - \sum_{j \neq i} \lambda_{ji}) + \frac{1}{mn} \frac{wabe^{-bt}}{(1 + ae^{-bt})^2} \Delta t + p - p\alpha_i(1 - \sum_{j \neq i} \lambda_{ji}). \quad (8)$$

In addition, to obtain the equilibrium prices efficiently, some notations are necessary to be introduced in advance. Let $\mathbf{B}^{n \times n} = (b_{ij})_{ij}$, where $b_{ij} = 2$ if $i = j$, and $b_{ij} = -\lambda_{ij}$ otherwise. Let

$$\begin{aligned} \mathbf{u}^{n \times 1} &= \left(\frac{U_i^{r-1}}{m} + c\alpha_i(1 - \sum_{j \neq i} \lambda_{ji}) + \frac{1}{mn} \frac{wabe^{-bt}}{(1 + ae^{-bt})^2} \Delta t \right)_{i \in N}, \\ \mathbf{v}^{n \times 1} &= (1 - \alpha_i(1 - \sum_{j \neq i} \lambda_{ji}))_{i \in N}. \end{aligned}$$

Meanwhile, let the equilibrium prices be a vector form $\mathbf{P}^{\text{ne}} = (p_i^{r, \text{ne}})_{i \in N}$. Then put Eq. (8) in matrix form, we get $\mathbf{B}\mathbf{P}^{\text{ne}} = \mathbf{u} + p\mathbf{v}$. Because $\sum_{j, j \neq i} \lambda_{ij} < 1$, $\forall i \in N$, we have $b_{ii} = 2 > 1 > |-\sum_{j \neq i} \lambda_{ij}| = |\sum_{j \neq i} b_{ij}|$, i.e., each diagonal element of \mathbf{B} is larger than the sum of other elements in the same row. Thus, \mathbf{B} is invertible and the equilibrium prices can be efficiently computed by

$$\mathbf{P}^{\text{ne}} = \mathbf{B}^{-1}(\mathbf{u} + p\mathbf{v}).$$

Based on above analysis results, we propose the following implications of the strategic behaviors of the operators in each round. For simplicity, we explain the implications with the help of Eq. (7) to make the readers better understand each operator's equilibrium strategy.

- (1) Because $\frac{\partial p_i^{r, \text{ne}}}{\partial p} > 0, \forall i \in N$, the equilibrium price changes in the same direction as the renting price does. It means if the joint venture raises (or decreases) the unit renting price p in the r -th round, then all operators will raise (or decrease) their unit service price simultaneously to promise their own profits.
- (2) Because $\frac{\partial p_i^{r, \text{ne}}}{\partial U_i^{r-1}} > \frac{\partial p_i^{r, \text{ne}}}{\partial U_j^{r-1}} > 0, \forall j \neq i$, the number of one operator's users has more influence on her own equilibrium price than the influence of other operators does.
- (3) Since $\frac{\partial p_i^{r, \text{ne}}}{\partial \alpha_i} \propto (c - p), \forall i \in N$, if $c - p < 0$ (i.e., $p > c$), the operator changes her equilibrium price with her own share in the opposite direction. In this situation, the joint venture could benefit from renting shared cars. Then if the operator has more share, she will care more about the her profit from the joint venture than the profit from users. So she will reduce the unit service price p_i to increase the quantity of demand, i.e., the renting times, for the joint venture. On the contrary, under the condition that $c - p > 0$ (i.e., $p < c$), the joint venture will not benefit from renting the shared cars. In fact, it is losing money for each renting. Then all operators only care about their profit from the end user market. So if an operator shares more in the joint venture, she will raise the unit service price to reduce the renting times and gain more from her each service.

(4) Since the equilibrium price (7) can be rewritten as

$$p_i^{r,ne} = \frac{1}{2+\lambda} \left(\lambda P + \frac{U_i^{r-1}}{m} + p - \lambda' \alpha_i (p - c) + \frac{1}{mn} \frac{ds(t)}{dt} \Big|_{t=(r-1)\Delta t + \frac{\Delta t}{2}} \Delta t \right),$$

we have $\frac{\partial p_i^{r,ne}}{\partial r} = \frac{1}{2+\lambda} \frac{1}{mn} \frac{d^2 s(t)}{dt^2} \Big|_{t=(r-1)\Delta t + \frac{\Delta t}{2}} \Delta t^2 = \eta \frac{d^2 s(t)}{dt^2} \Big|_{t=(r-1)\Delta t + \frac{\Delta t}{2}}$, where $\eta = \frac{1}{2+\lambda} \frac{1}{mn} \Delta t^2 > 0$. Because of the concavity and convexity of $s(t)$, we could obtain that there exists a critical round r^* such that if $r \leq r^*$, then $\frac{\partial p_i^{r,ne}}{\partial r} > 0$ and if $r \geq r^*$, then $\frac{\partial p_i^{r,ne}}{\partial r} < 0$. It changes after the explosive growth of the total number of users. When $\frac{\partial p_i^{r,ne}}{\partial r} > 0$, the operators are facing a flood of new users. They prefer to raise their unit service price and don't worry about losing a few users. On the contrary, when the user scale of the whole market tends to be stable, the slow growth of the number of new users forces the operators to reduce their service prices to retain their own current users and attract other users.

3.2 First Stage: Aggregate Profit Maximization

In this section, we discuss the first stage of the Stackelberg game for the joint venture in the r -th round. Here, we focus on the case with $\lambda_{ij} = \lambda, \forall i, j \in N, j \neq i$. The analysis of general cases with arbitrary λ_{ij} is similar.

In each round, the joint venture decides the unit renting price p and the share α_i for each operator i based on the current demand in the market and the equilibrium strategies of all operators. Different from the operators, the joint venture aims to maximize the aggregate profit of the entire market in each round. This profit is given by

$$AP = \sum_{i \in N} (p_i - c) (U_i^{r-1} - mp_i + m\lambda \sum_{j \neq i} p_j + \frac{1}{n} \frac{wabe^{-bt}}{(1 + ae^{-bt})^2} \Delta t) \theta.$$

By solving $\frac{\partial AP}{\partial p_i} = 0$, the joint venture obtains the aggregate profit which is maximized at

$$p_i^r = \frac{c}{2} + \frac{\lambda' U_i^{r-1} + \lambda \sum_{j \in N} U_j^{r-1}}{(1 + \lambda) 2m\lambda'} + \frac{1}{2mn\lambda'} \frac{wabe^{-bt}}{(1 + ae^{-bt})^2} \Delta t \quad (9)$$

In order to make sure that the equilibrium renting price leads to the maximal aggregate profit, we combine Eqs. (7) and (9) for all $i \in N$ together with condition of $\sum_{i \in N} \alpha_i = \alpha$ to calculate the value of p and α_i under Stackelberg equilibrium as follows:

$$p^{r,ne} = c + \frac{(n-1)\lambda(\sum_{i \in N} U_i^{r-1} + \frac{wabe^{-bt}}{(1+ae^{-bt})^2} \Delta t - mn\lambda'c)}{2m\lambda'(n-\alpha\lambda')} \quad (10)$$

$$\alpha_i^{r,ne} = \frac{\alpha}{n} + \frac{(n-\lambda'\alpha)(U_i^{n-1} - \frac{\sum_{i \in N} U_i^{r-1}}{n})}{(n-1)(1+\lambda)(\sum_{i \in N} U_i^{r-1} + \frac{wabe^{-bt}}{(1+ae^{-bt})^2} \Delta t - mn\lambda'c)} \quad (11)$$

In other words, as long as the joint venture sets p and α_i as Eqs. (10) and (11), the total profit of all operators will coincide with the aggregate profit of the whole market. From Eq. (11) we find that each operator’s share of the joint venture is related to the number of her current users. Specifically, the difference between one’s share and the average share is linear to the difference between the number of her users and the average number of the total users.

4 Numerical Results

In this section, we conduct numerical experiments to verify our results and analysis. We conduct two sets of experiments with three operators. Without loss of generality, we set the three parameters w, a, b in Eq. (1) as 10000, 500 and 0.13 respectively. The length of each round (i.e., Δt) is set as 1. In addition, we set $m = 3, c = 5, \lambda_{ij} = 0.4, \forall i, j \in \{1, 2, 3\}, j \neq i$.

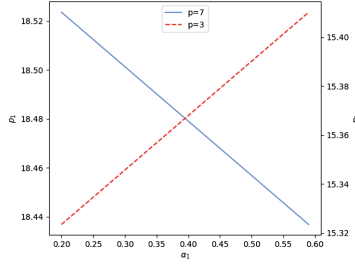


Fig. 2. The influence of α_1 on the unit service price p_1 with different renting price p .

In the first experiment, we show the influence of the share of operator 1 (i.e., α_1) on price p_1 given different renting price p . We set the number of their current users $U_1 = 28, U_2 = 22, U_3 = 40$. The share of operator 1 α_1 ranges from 0.2 to 0.6 with an increment of 0.01 while α_2 and α_3 are fixed at 0.2. Figure 2 shows the change of operator 1’s unit service price under different α_1 and different p at the 24th round. The solid line is corresponding to the left vertical axis and the dashed line is corresponding to the right one. As we can see, when $p = 7$, i.e., the unit renting price is larger than the unit cost, the service price of operator 1 decreases as her share α_1 increases. When $p = 3$, i.e., the unit renting price is lower than the unit cost, p_1 increases as α_1 increases. It matches with what we discussed before. That is, if the joint venture could benefit from renting the shared cars,

operators would rather reduce their service price to increase the profit from the joint venture. On the contrary, if the joint venture can not benefit from renting, then operators prefer to raise price to reduce the renting times.

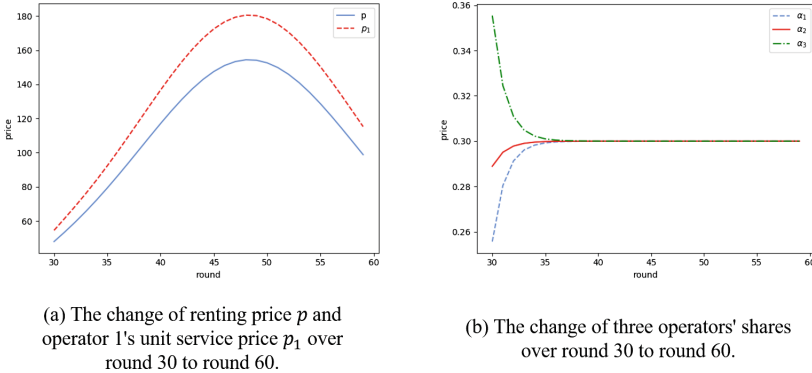


Fig. 3. The change of prices and shares over round 30 to round 60.

In the second experiment, we show the changes of renting price, service price and share over time. Specifically, we set the number of these three operators' users at the end of 29th round as 28, 22 and 40 respectively. Figure 3(a) shows the changes of renting price p and operator 1's unit service price p_1 over the 30th round to the 60th round. As we can see, p_1 is always larger than p and these two lines has the same trend of change, which is more related to the growth rate of the number of users in Fig. 1(b). In the first half of the period, the whole market is experiencing an explosive growth. As a result, the joint venture increases the renting price for more profit and operator 1 raises her service price as well. Then as the number of users grows slowly, they begin to reduce the price to keep users for the long term profit. Figure 3(b) shows the changes of three operators' shares over time. As time goes by, three operators share the same profit from the joint venture gradually. Because the share is related to the number of their own users, the same share means they have almost the same user scale at the end. In other words, they divide up the market equally.

5 Conclusion

In this paper, we present a novel business system by introducing a joint venture to the electric car sharing market to reduce the competitive operators' risk and the social resource consumption. We model the number of the users has a S-shaped growth over time, inspired from the bicycle sharing market, and then discretize the time into infinite rounds to facilitate the analysis. In each round, we apply the game theory approach to model the business system as a two-stage Stackelberg game, in which the joint venture is the leader and the operators are

the followers. Our main contributions propose the equilibrium strategies of the leader and the followers in each round and reveal a possible way to solve some sharp problems, such as the redundancies in suppliers, in bicycle sharing market.

In this paper, we assume that the joint venture rents the cars to all operators at the same price. However, the renting price may be different to each operator. Thus, we will study a model, in which the joint venture sets the different renting prices to the operators based on their different reputations. In such a model, how to design a protocol to compute each operator's reputation is a crux. In addition, we also find some users might stick with the operator they used to in practice, even though the service price of this operator increases. So, choosing an appropriate function, instead of the linear one, to characterize the relation between the number of the users loss and the unit service price, is very important.

References

1. The Bike-Share Oversupply in China: Huge Piles of Abandoned and Broken Bicycles, The Atlantic. <https://www.theatlantic.com/photo/2018/03/bike-share-oversupply-in-china-huge-piles-of-abandoned-and-broken-bicycles/556268/>. Accessed 23 Feb 2019
2. The Rise and Fall of Dockless Bike Sharing in Dallas. <https://www.texasmonthly.com/news/rise-fall-dockless-bike-sharing-dallas/>. Accessed 23 Feb 2019
3. Deng, X.T., Wang, J.P., Wang, J.T.: How to design a common telecom infrastructure for competitors to be individually rational and collectively optimal. *IEEE J. Sel. Areas Commun.* **35**(3), 736–750 (2017)
4. How China's Bike-Sharing Startup Ofo Went From Tech Darling To Near Bankruptcy. <https://www.forbes.com/sites/ywang/2018/12/20/how-chinas-bike-sharing-startup-ofo-went-from-tech-darling-to-near-bankruptcy>. Accessed 23 Feb 2019
5. Gao, Y.: Telecom firms establish base station JV, China Daily. http://www.chinadaily.com.cn/business/2014-07/12/content_17740185.htm. Accessed 23 Feb 2019
6. Guidance on promoting the healthy development of small and micro car rental, The Ministry of Transport and The Ministry of Housing and Urban-Rural Development. http://xxgk.mot.gov.cn/jigou/ysfws/201708/t20170807_2978817.html. Accessed 23 Feb 2019
7. Kucharavy, D., Guio, R.D.: Application of S-shaped curves. *Procedia Eng.* **9**(none), 559–572 (2011)
8. Ma, L., Zhang, X., Wang, G.S.: Identifying the reasons why users in China recommend bike apps. *Int. J. Mark. Res.* **59**(6), 767–786 (2017)
9. Pohjola, M.: Nash and Stackelberg solutions in a differential game model of capitalism. *J. Econ. Dyn. Control.* **6**(none), 173–186 (1983)
10. Singh, N., Vives, X.: Price and quantity competition in a differentiated duopoly. *RAND J. Econ.* **15**(4), 546–554 (1984)
11. Report on Bicycle Sharing Market, QuestMobile. <https://www.questmobile.com.cn/research/report-historical/57>. Accessed 23 Feb 2019
12. First half of 2018 China bicycle sharing industry research report. <https://baijiahao.baidu.com/s?id=1603881982645198832&wfr=spider&for=pc>. Accessed 5 Jan 2019
13. China Development Report for Time-sharing Rental of New Energy Vehicle, Roland Berger. <https://www.rolandberger.com/zh/Publications/>. Accessed 5 Jan 2019



Constructing Three Completely Independent Spanning Trees in Locally Twisted Cubes

Kung-Jui Pai¹(✉), Ruay-Shiung Chang², Jou-Ming Chang², and Ro-Yu Wu³

¹ Department of Industrial Engineering and Management,
Ming Chi University of Technology, New Taipei City, Taiwan
`poter@mail.mcut.edu.tw`

² Institute of Information and Decision Sciences,
National Taipei University of Business, Taipei, Taiwan
`{rschang,spade}@ntub.edu.tw`

³ Department of Industrial Management,
Lunghwa University of Science and Technology, Taoyuan, Taiwan
`eric@mail.lhu.edu.tw`

Abstract. For the underlying graph G of a network, k spanning trees of G are called completely independent spanning trees (CISTs for short) if they are mutually inner-node-disjoint. It has been known that determining the existence of k CISTs in a graph is an NP-hard problem, even for $k = 2$. Accordingly, researches focused on the problem of constructing multiple CISTs in some famous networks. Pai and Chang [28] proposed a unified approach to recursively construct two CISTs with diameter $2n - 1$ in several n -dimensional hypercube-variant networks for $n \geq 4$, including locally twisted cubes LTQ_n . Later on, they provided a new construction for LTQ_n and showed that the diameter of two CISTs can be reduced to $2n - 2$ if $n = 4$ (and thus is optimal) and $2n - 3$ if $n \geq 5$. In this paper, we intend to construct more CISTs of LTQ_n . We develop a novel tree searching algorithm, called two-stages tree-searching algorithm, to construct three CISTs of LTQ_6 and show that the three CISTs of the high-dimensional LTQ_n for $n \geq 7$ can be constructed by recursion. The diameters of three CISTs for LTQ_n we constructed are 9, 12 and 14 when $n = 6$, and are $2n - 3$, $2n - 1$ and $2n + 1$ when $n \geq 7$.

Keywords: Completely independent spanning trees · Interconnection networks · Locally twisted cubes · Diameter

1 Introduction

Let $k \geq 2$ and T_1, T_2, \dots, T_k be spanning trees of a graph $G = (V, E)$. A node is a *leaf* in a tree T_i if it has degree 1, and an *inner-node* otherwise. Two spanning trees T_i and T_j are *edge-disjoint* if they share no common edge, and are *inner-node-disjoint* provided the paths joining any two nodes $u, v \in V$ in both trees

have no node in common except for u and v . The spanning trees T_1, T_2, \dots, T_k are *completely independent spanning trees* (CISTs for short) if they are pairwise inner-node-disjoint (and thus are edge-disjoint, see Theorem 1).

Constructing CISTs in networks has applications on fault-tolerant routing and secure message transmission. Hasunuma [13] showed that determining if there exist k CISTs in a graph is NP-hard even for $k = 2$. Hence, finding sufficient conditions for graphs that admit multiple CISTs has been investigated in [1, 2, 9, 14, 19]. Especially, the degree-based condition such as Dirac's condition [1] and Ore's condition [9]. Péterfalvi [32] showed that, for any $k \geq 2$, there exists a k -connected graph which does not possess two CISTs. This disproved a conjecture posed by Hasunuma [13], which states that there exist k CISTs in a $2k$ -connected graph (see also [31] for more counterexamples). Also, it has been confirmed by construction that certain classes of graphs possess two CISTs, e.g., 4-connected maximal planar graphs [13], the Cartesian product of any 2-connected graphs [15], and 4-regular chordal rings [4, 31]. Moinet et al. [27] recently investigated the problem of constructing CISTs in ad-hoc networks under a practical approach and showed through simulation results that more CISTs can be found when the network density is sufficiently high. In addition, graphs possessing more CISTs can be found in some regular graphs [8], the underlying graphs of line digraphs [12], partial k trees [26], complete graphs, complete bipartite graphs and complete tripartite graphs [30].

In particular, Pai and Chang [28] proposed a unified approach to recursively construct two CISTs in several hypercube-variant networks, including hypercubes, locally twisted cubes, crossed cubes, parity cubes, and Möbius cubes (see also [7] for another constructing scheme of crossed cubes and a special bijection connection networks). The *diameter* of a graph G is the greatest distance between any pair of nodes in G . Pai and Chang [28] showed that the diameter of CISTs for their construction in hypercube-variant networks is $2n - 1$ for $n \geq 4$. Afterward, Pai and Chang [29] also provided a new construction scheme for locally twisted cubes and showed that the diameter of two CISTs can be reduced to $2n - 2$ if $n = 4$ (and thus is optimal) and $2n - 3$ if $n \geq 5$. In this paper, we continue to explore the problem of constructing more CISTs on locally twisted cubes. The following two characterizations are important for studying CISTs.

Theorem 1 (See [12, Theorem 2.1]). *A set of spanning trees T_1, T_2, \dots, T_k are CISTs in a graph $G = (V, E)$ if and only if they are edge-disjoint in G and for any $v \in V$, there is at most one T_i such that v is an inner-node.*

Theorem 2 (See [1, Theorem 2.3]). *A graph $G = (V, E)$ admits k CISTs if and only if there is a partition of V into V_1, V_2, \dots, V_k , which is called a k -CIST-partition, such that the following hold:*

- (i) *for $i \in \{1, 2, \dots, k\}$, the subgraph of G induced by V_i , denoted by $G[V_i]$, is connected;*
- (ii) *for distinct $i, j \in \{1, 2, \dots, k\}$, the bipartite graph with bipartition $V_i \cup V_j$ and edge set $\{(x, y) \in E(G) : x \in V_i, y \in V_j\}$, denoted by $B(V_i, V_j, G)$, has no tree component.*

Note that the later condition in Theorem 2 is equivalent to the condition that every connected component $H = (V_H, E_H)$ of $B(V_i, V_j, G)$ satisfies $|E_H| \geq |V_H|$.

For example, Fig. 1 shows two CISTs of the locally twisted cube LTQ_4 (defined later in Sect. 2 for LTQ_n). It is easy to verify that both T_1 and T_2 are edge-disjoint, and every node of LTQ_4 is a leaf either in T_1 or T_2 . Thus, by Theorem 1, the two spanning tree shown in Fig. 1 are indeed CISTs.

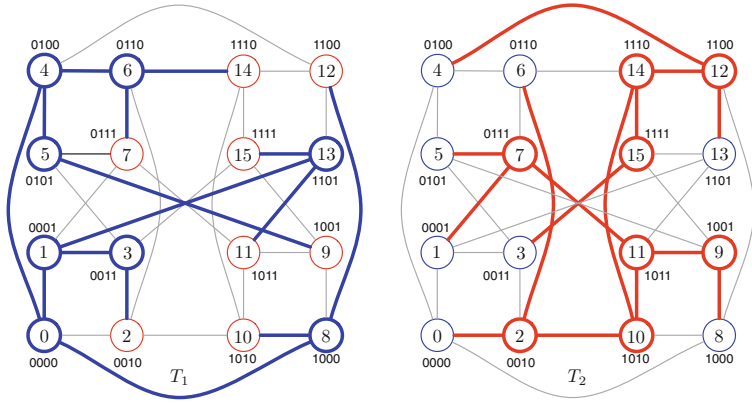


Fig. 1. Two CISTs of LTQ_4 , where thick lines indicate edges of spanning trees.

In this paper, we develop a novel tree-searching algorithm to find a 3-CIST-partition of nodes for locally twisted cube LTQ_6 . According to Theorem 2, we can obtain the desired three CISTs of LTQ_6 . Then, using the result as induction base, we show that three CISTs for high-dimensional LTQ_n can be constructed by recursion. Consequently, we acquire the following result.

Theorem 3. *The locally twisted cube LTQ_n admits three CISTs with diameters 9, 12 and 14 when $n = 6$, and with diameters $2n - 3$, $2n - 1$ and $2n + 1$ when $n \geq 7$.*

2 Locally Twisted Cubes

Let \oplus denote the modulo 2 addition. For $n \geq 2$, a node v in a locally twisted cube is labeled by using a binary string $v = v_{n-1}v_{n-2} \cdots v_0$, where $v_i \in \{0, 1\}$ for $0 \leq i \leq n - 1$. For conciseness, the label of v is changed to its decimal representation if it needs. Let G^x be the graph obtained from a labeled graph G by prefixing the binary string of every node with x . The n -dimensional locally twisted cube LTQ_n is the labeled graph with the following recursive fashion (see [38]):

- (1) LTQ_1 is the complete graph on two nodes labeled by 0 and 1. LTQ_2 is a graph consisting of four nodes with labels 00, 01, 10, 11 together with four edges (00, 01), (01, 11), (11, 10), and (10, 00).
- (2) For $n \geq 3$, LTQ_n is composed of two subcubes LTQ_{n-1}^0 and LTQ_{n-1}^1 such that each node $x = 0x_{n-2}x_{n-3} \cdots x_0 \in V(LTQ_{n-1}^0)$ is connected with the

node $y = 1(x_{n-2} \oplus x_0)x_{n-3} \cdots x_0 \in V(LTQ_{n-1}^1)$ by an edge, where x and y are called the $(n - 1)$ -neighbors to each other, and denote as $N_{n-1}(x) = y$ or $N_{n-1}(y) = x$.

From the above definition, it is clear that LTQ_n is an n -regular graph, and the binary strings of any two adjacent nodes in LTQ_n differ in at most two successive bits. For the convenience of referring the adjacency of nodes in the next section, we list all neighbors of nodes of LTQ_6 in the following table.

Table 1. The neighbors of nodes in LTQ_6

x	$N_0(x)$	$N_1(x)$	$N_2(x)$	$N_3(x)$	$N_4(x)$	$N_5(x)$	x	$N_0(x)$	$N_1(x)$	$N_2(x)$	$N_3(x)$	$N_4(x)$	$N_5(x)$
0 000000	1	2	4	8	16	32	32 100000	33	34	36	40	48	0
1 000001	0	3	7	13	25	49	33 100001	32	35	39	45	57	17
2 000010	3	0	6	10	18	34	34 100010	35	32	38	42	50	2
3 000011	2	1	5	15	27	51	35 100011	34	33	37	47	59	19
4 000100	5	6	0	12	20	36	36 100100	37	38	32	44	52	4
5 000101	4	7	3	9	29	53	37 100101	36	39	35	41	61	21
6 000110	7	4	2	14	22	38	38 100110	39	36	34	46	54	6
7 000111	6	5	1	11	31	55	39 100111	38	37	33	43	63	23
8 001000	9	10	12	0	24	40	40 101000	41	42	44	32	56	8
9 001001	8	11	15	5	17	57	41 101001	40	43	47	37	49	25
10 001010	11	8	14	2	26	42	42 101010	43	40	46	34	58	10
11 001011	10	9	13	7	19	59	43 101011	42	41	45	39	51	27
12 001100	13	14	8	4	28	44	44 101100	45	46	40	36	60	12
13 001101	12	15	11	1	21	61	45 101101	44	47	43	33	53	29
14 001110	15	12	10	6	30	46	46 101110	47	44	42	38	62	14
15 001111	14	13	9	3	23	63	47 101111	46	45	41	35	55	31
16 010000	17	18	20	24	0	48	48 110000	49	50	52	56	32	16
17 010001	16	19	23	29	9	33	49 110001	48	51	55	61	41	1
18 010010	19	16	22	26	2	50	50 110010	51	48	54	58	34	18
19 010011	18	17	21	31	11	35	51 110011	50	49	53	63	43	3
20 010100	21	22	16	28	4	52	52 110100	53	54	48	60	36	20
21 010101	20	23	19	25	13	37	53 110101	52	55	51	57	45	5
22 010110	23	20	18	30	6	54	54 110110	55	52	50	62	38	22
23 010111	22	21	17	27	15	39	55 110111	54	53	49	59	47	7
24 011000	25	26	28	16	8	56	56 111000	57	58	60	48	40	24
25 011001	24	27	31	21	1	41	57 111001	56	59	63	53	33	9
26 011010	27	24	30	18	10	58	58 111010	59	56	62	50	42	26
27 011011	26	25	29	23	3	43	59 111011	58	57	61	55	35	11
28 011100	29	30	24	20	12	60	60 111100	61	62	56	52	44	28
29 011101	28	31	27	17	5	45	61 111101	60	63	59	49	37	13
30 011110	31	28	26	22	14	62	62 111110	63	60	58	54	46	30
31 011111	30	29	25	19	7	47	63 111111	62	61	57	51	39	15

As to locally twisted cubes, constructing rooted spanning trees has devoted many kinds of research. Hsieh and Tu [17] proposed an algorithm to construct n edge-disjoint spanning trees rooted at a particular node 0 in LTQ_n . Later on, Lin et al. [22] proved that Hsieh and Tu's spanning trees are indeed independent spanning trees (ISTs for short), i.e., all spanning trees are rooted at the same node r and for any other node $v(\neq r)$, all paths from v to r in these trees are inner-node-disjoint. Shortly afterward, Liu et al. [23] pointed out that LTQ_n fails to be node-transitive for $n \geq 4$ and proposed an algorithm for constructing

n ISTs rooted at an arbitrary node in LTQ_n . Recently, Chang et al. [6] proposed a fully parallelized scheme to construct n ISTs rooted at an arbitrary node in LTQ_n . Except above, many results in the previous research on LTQ_n can be found in the literature, e.g., the studies of fault-tolerant Hamiltonicity and related problems [18, 20, 25, 35, 36], tree and mesh embeddings [3, 11, 21, 24], restricted connectivity [5, 10, 16], and diagnosability [33, 34, 37].

3 An Algorithm to Find Three CISTs of LTQ_6

In this section, we develop a tree searching algorithm, called two-stages tree-searching algorithm (abbreviated \mathbf{TS}^2), to find a 3-CIST-partition of $V(LTQ_6)$. Clearly, $|V(LTQ_6)| = 2^6 = 64$. We are looking forward to obtaining a near equalized partition (even though this is not necessary). Let $n = |V_1| = |V_2| = |V_3| - 1 = \lfloor \frac{64}{3} \rfloor = 21$. It is as the name suggested that \mathbf{TS}^2 has two stages for tree searching, where the first stage attempts to find nodes of V_1 such that $LTQ_6[V_1]$ induces a minimally connected subgraph (i.e., a tree) in LTQ_6 , and the second stage then finds nodes of V_2 such that $LTQ_6[V_2]$ also induces a tree. To avoid larger diameters in the resulting trees, we employ the breadth-first search as the searching strategy in each process of \mathbf{TS}^2 . After the two candidate sets V_1 and V_2 being found out, we need to check the connectedness of $LTQ_6[V_3]$ and all the remaining conditions between two of the sets (i.e., the condition (ii) in Theorem 2).

For the convenience of description, we use arrays $V_i[1..n]$ for $i \in \{1, 2, 3\}$ to represent the three sets in a 3-CIST-partition. Also, for flexibility, we allow the use of set-related operations in $V_j[1..i]$ for $1 \leq i \leq n$ and $1 \leq j \leq 3$. The following two procedures are the expansions for finding the next feasible node in each stage of \mathbf{TS}^2 . Initially, we set $V_1[1] = 0$ and perform Expand-V1(1).

Procedure Expand-V1(i)

```

1 begin
2   if  $i < n$  then
3     for each  $v \in N_{LTQ_6}(V_1[i])$  do
4       if  $v \notin V_1[1..i-1]$  and no node of  $V_1[1..i-1]$  is adjacent to  $v$  then
5         Enqueue( $Q_1, v$ );
6       while  $Q_1$  is not empty do
7          $u \leftarrow$  Dequeue( $Q_1$ );
8         if there is only one node of  $V_1[1..i]$  such that it is adjacent to  $u$  and
           the maximum degree of  $LTQ_6[V_1[1..i] \cup \{u\}]$  is no more than 4 then
9            $V_1[i+1] \leftarrow u$ ;
10          call Expand-V1( $i+1$ );
11        else
12          set  $\bar{V}_1 \leftarrow V \setminus V_1$  and  $x$  be any node in  $\bar{V}_1$ ;
13          set  $V_2[1] \leftarrow x$ ;
14          call Expand-V2(1);

```

Procedure Expand-V2(i)

```

1 begin
2   if  $i < n$  then
3     for each  $v \in N_{LTQ_6[\bar{V}_1]}(V_2[i])$  do
4       if  $v \notin V_2[1..i-1]$  and no node of  $V_2[1..i-1]$  is adjacent to  $v$  then
5          $\text{Enqueue}(Q_2, v)$ ;
6     while  $Q_2$  is not empty do
7        $u \leftarrow \text{Dequeue}(Q_2)$ ;
8       if there is only one node of  $V_2[1..i]$  such that it is adjacent to  $u$  and
9         the maximum degree of  $LTQ_6[V_2[1..i] \cup \{u\}]$  is no more than 4 then
10         $V_2[i+1] \leftarrow u$ ;
11        call Expand-V2( $i+1$ );
12   else
13      $V_3 \leftarrow \bar{V}_1 \setminus V_2$ ;
14     if  $LTQ_6[V_3]$  is a tree or unicyclic graph and  $|E(H)| \geq |V(H)|$  for each
15       component  $H$  in  $B(V_j, V_k, LTQ_6)$  for  $j, k \in \{1, 2, 3\}$  with  $j \neq k$  then
16        $\text{output } V_1, V_2$  and  $V_3$  as a 3-CIST-partition;

```

In the above two procedures, statements on lines 3–10 are similar and are used to find the candidate sets V_1 and V_2 , respectively. For $j = 1, 2$, the **for**-loop in lines 3–5 filter the neighbors of the currently expanded node $V_j[i]$ so that they have a chance to become candidates waiting in the queue Q_j to be expanded. The statement of line 8 in the **while**-loop guarantees that the next expanded node u must be a leaf in the current tree $LTQ_6[V_j[1..i] \cup \{u\}]$. Also, the second condition requires that the maximum degree in the current tree is no more than 4 because that LTQ_6 is a regular graph of degree 6 and every subgraph induced by V_i for $i \in \{1, 2, 3\}$ must be connected. The statement of line 13 in the procedure Expand-V2 checks the connectedness of $LTQ_6[V_3]$, where a *unicyclic graph* is defined to be a connected graph with exactly one cycle, which can be recognized in linear time. In addition, checking the condition $|E(H)| \geq |V(H)|$ for each component H in $B(V_j, V_k, LTQ_6)$ can be done by a DFS or BFS search. Therefore, by Theorem 2, the algorithm **TS**² consisting of the two procedures described above can be used to obtain a 3-CIST-partition of $V(LTQ_6)$ if it exists. Fortunately, we acquire the desired partition of $V(LTQ_6)$ generated by **TS**² as follows:

$$\begin{aligned}
 V_1 &= \{0, 1, 2, 4, 7, 13, 10, 18, 20, 36, 31, 55, 15, 61, 42, 44, 43, 58, 56, 57, 33\}, \\
 V_2 &= \{3, 5, 29, 28, 45, 24, 30, 8, 22, 62, 40, 46, 41, 37, 49, 21, 48, 19, 50, 11, 59\}, \\
 V_3 &= \{6, 9, 12, 14, 16, 17, 23, 25, 26, 27, 32, 34, 35, 38, 39, 47, 51, 52, 53, 54, 60, 63\},
 \end{aligned}$$

where V_1 and V_2 are listed according to the order in the queues Q_1 and Q_2 , respectively. By a lengthy checking the adjacency described in Table 1, we can

verify that each $LTQ_6[V_i]$ for $i \in \{1, 2, 3\}$ is connected as shown in Fig. 2. As we can see, $LTQ_6[V_1]$ and $LTQ_6[V_2]$ are trees and $LTQ_6[V_3]$ is a unicyclic graph.

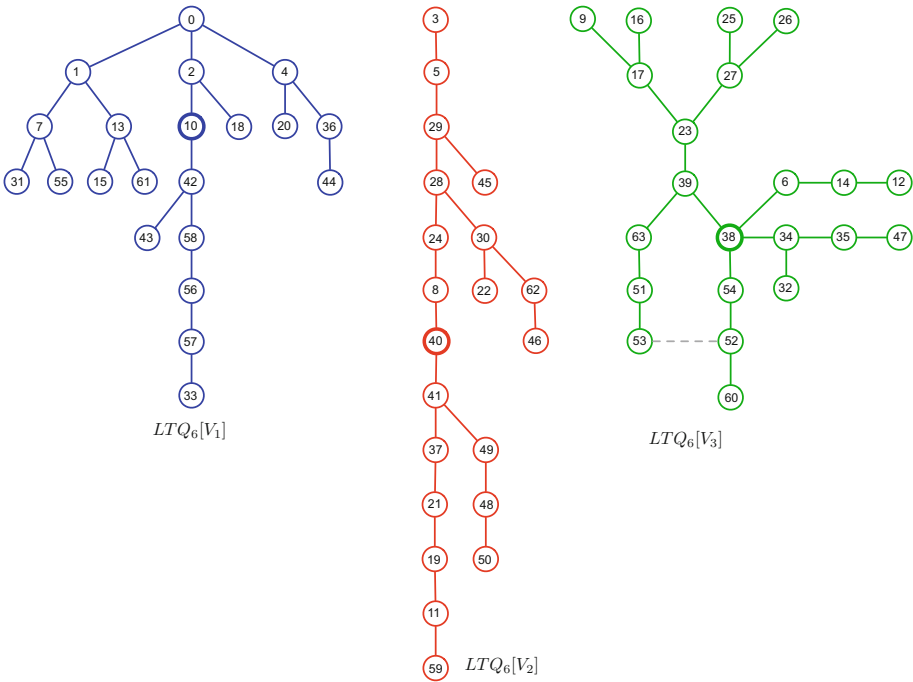


Fig. 2. Three connected graphs $LTQ_6[V_i]$ for $i \in \{1, 2, 3\}$.

Figure 3 shows three bipartite graphs $B(V_i, V_j, LTQ_6)$ for $i, j \in \{1, 2, 3\}$ with $i \neq j$. $B(V_1, V_2, LTQ_6)$ contains two components, one has cycle $(55, 49, 61, 59, 55)$ and the other has cycle $(42, 40, 44, 46, 42)$. $B(V_2, V_3, LTQ_6)$ contains one component with cycle $(41, 25, 24, 26, 30, 14, 46, 47, 41)$. $B(V_1, V_3, LTQ_6)$ contains one component with two cycles $(15, 63, 57, 9, 15)$ and $(0, 16, 20, 52, 36, 32, 0)$. Since every bipartite graph has no tree component, by Theorem 2, V_1, V_2 and V_3 form a 3-CIST-partition of $V(LTQ_6)$.

Based on the above partition, we can construct three CISTs of LTQ_6 in the following way. For each $i \in \{1, 2, 3\}$, we choose V_i as the inner-nodes of T_i , such that $LTQ_6[V_i]$ forms a subtree of T_i , and particularly, we remove the edge $(52, 53)$ in $LTQ_6[V_3]$ (see the dashed line in Fig. 2). Then, for each node $u \in V_i$, we take nodes $v \in V_j$ where $j \in \{1, 2, 3\} \setminus \{i\}$ as leaves to join the inner-node u in T_i if (u, v) is an edge of $B(V_i, V_j, LTQ_6)$ and its color is the same as u 's color in Fig. 3. It is easy to inspect that every node $v \in V_j$ is exactly joined to an inner-node of T_i , and thus the resulting graph is a spanning tree. Since $|E(LTQ_6)| = 192$ and each spanning tree requires 63 edges, except the edge $(52, 53)$, another two unused edges are $(22, 23)$ and $(18, 26)$ (which are drawn

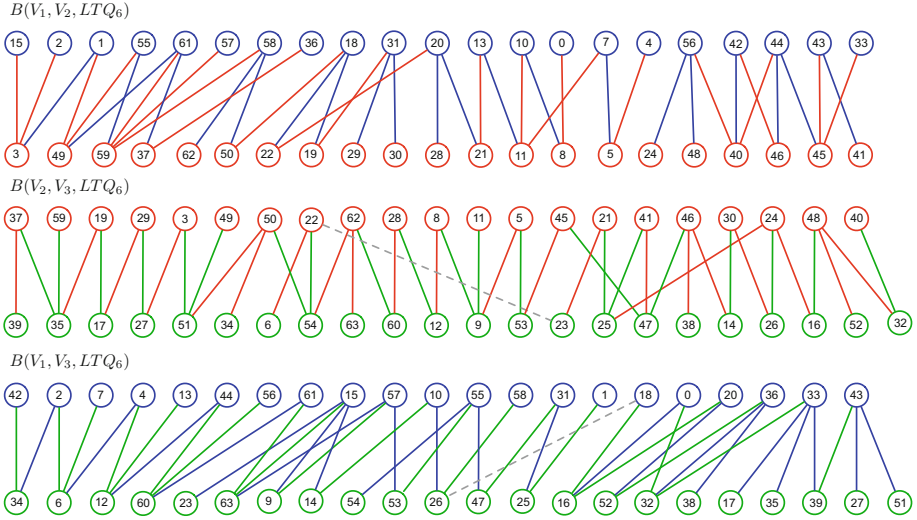


Fig. 3. Bipartite graphs $B(V_i, V_j, LTQ_6)$ for $i, j \in \{1, 2, 3\}$ with $i \neq j$.

by dashed lines in Fig. 3). Consequently, the diameters of spanning trees T_1 , T_2 and T_3 we constructed are 12, 14 and 9, respectively. We summarize the above construction in the following lemma.

Lemma 1. *For LTQ_6 , it admits three CISTs with diameters 9, 12 and 14.*

4 Constructing Three CISTs on High-Dimensional LTQ_n

In this section, we give a recursive construction of three CISTs in locally twisted cubes LTQ_n for $n \geq 7$. Recall that G^0 (resp. G^1) denotes the labeled graph G with a prefix symbol 0 (resp. 1) in every node. In [28], Pai and Chang proved the following result.

Theorem 4 (See [28, Theorem 4]). *Let G_{n-1} be the $(n-1)$ -dimensional variant hypercube for $n \geq 5$ and suppose that T_1 and T_2 are two CISTs of G_{n-1} . For $i \in \{1, 2\}$, let \hat{T}_i be a spanning tree of G_n constructed from T_i^0 and T_i^1 by adding an edge $(u_i, v_i) \in E(G_n)$ to connect two inner-nodes $u_i \in V(T_i^0)$ and $v_i \in V(T_i^1)$. Then, \hat{T}_1 and \hat{T}_2 are two CISTs of G_n .*

We can easily generalize the following property by using the same proof technique of Theorem 4.

Corollary 1. *Let G_{n-1} be the $(n-1)$ -dimensional variant hypercube for $n \geq 5$ and suppose that T_i for $1 \leq i \leq k$ (where $2 \leq k < n$) are k CISTs of G_{n-1} . Let \hat{T}_i be a spanning tree of G_n constructed from T_i^0 and T_i^1 by adding an edge $(u_i, v_i) \in E(G_n)$ to connect two inner-nodes $u_i \in V(T_i^0)$ and $v_i \in V(T_i^1)$. Then, \hat{T}_i for $1 \leq i \leq k$ are k CISTs of G_n .*

In the above property, two nodes $u_i \in V(T_i^0)$ and $v_i \in V(T_i^1)$ are called the *port nodes* of the trees T_i^0 and T_i^1 , respectively. Also, the edge $(u_i, v_i) \in E(G_n)$ is called the *bridge* of \hat{T}_i for the construction. Obviously, choosing an appropriate pair $\{u_i, v_i\}$ as port nodes is important because it is the key factor in determining the diameter of the resulting tree. The *eccentricity* of a node v in a graph G , denote by $e_G(v)$, is defined to be the maximum distance between v and any other node in G . Hence, the maximum eccentricity is the graph diameter, denoted as $\text{diam}(G)$. Also, a set of nodes with the minimum eccentricity is called the *center*. A well-known result is that the center in a tree T consists of either a singleton if $\text{diam}(T)$ is even or two adjacent nodes otherwise. Moreover, by Corollary 1, the diameter of the resulting tree can be calculated as follows:

$$\text{diam}(\hat{T}_i) = e_{T_i^0}(u_i) + e_{T_i^1}(v_i) + 1 \text{ for } i \in \{1, 2, \dots, k\} \tag{1}$$

We are now in a position to prove the main result.

Proof of Theorem 3. By Lemma 1, there exist three CISTs in LTQ_6 with the diameters 9, 12 and 14, respectively. We now consider LTQ_n for $n \geq 7$. Suppose that T_i for $i \in \{1, 2, 3\}$ are CISTs of LTQ_{n-1} . Note that the $(n-1)$ -dimensional neighbor of a node $x \in V(LTQ_{n-1})$ in LTQ_n is defined to be $N_{n-1}(x) = x + 2^{n-1}$ when x is even (in this case, x is called an even node). Hence, for each $i \in \{1, 2, 3\}$, we can choose an even node $x \in V(T_i^0)$ and its neighbor $N_{n-1}(x) \in V(T_i^1)$ as port nodes to construct CISTs of LTQ_n . For instance, to construct three CISTs of LTQ_7 , we choose nodes 10, 40 and 38 as port nodes of T_1^0, T_2^0 and T_3^0 , respectively (see the nodes drawn by thick lines in Fig. 2, where all leaves are omitted in the drawing). Similarly, we choose nodes $74 (= 10 + 2^6)$, $104 (= 40 + 2^6)$ and $102 (= 38 + 2^6)$ as port nodes of T_1^1, T_2^1 and T_3^1 , respectively. Then, by adding the bridges $(10, 74)$, $(40, 104)$ and $(38, 102)$, we yield the desired three CISTs in LTQ_7 . Since $e_{T_1}(10) = 6$, $e_{T_2}(40) = 7$ and $e_{T_3}(38) = 5$, by Eq. (1), we have $\text{diam}(\hat{T}_1) = 13$, $\text{diam}(\hat{T}_2) = 15$ and $\text{diam}(\hat{T}_3) = 11$.

In general, we can choose the pair $\{10, 10 + 2^{n-1}\}$, $\{40, 40 + 2^{n-1}\}$ and $\{38, 38 + 2^{n-1}\}$ as port nodes for building three CISTs of LTQ_n by using induction on n . Then, by Lemma 1, Corollary 1, and Eq. (1), the result follows. \square

5 Concluding Remarks

In this paper, we construct three CISTs of LTQ_n . A novel tree searching algorithm, called two-stages tree-searching algorithm, is developed for constructing three CISTs of LTQ_6 . Moreover, we show that the three CISTs of the high-dimensional LTQ_n for $n \geq 7$ can be constructed by recursion. Consequently, the diameters of three CISTs we constructed are with diameters 9, 12 and 14 when $n = 6$, and with diameters no more than $2n + 1$ when $n \geq 7$. In future work, we will try to construct more CISTs for hypercube-variant networks.

As aforementioned, there exists a parallelized scheme to construct ISTs rooted at an arbitrary node in LTQ_n (see [6]). By contrast, all known existed algorithms for constructing CISTs of hypercube-variant networks are developed in a recursive fashion, and thus are hard to be parallelized. It seems very attractive to develop a parallelized and non-recursive algorithm for constructing CISTs in hypercube-variant networks.

Acknowledgments. This research was partially supported by MOST grants 107-2221-E-131-011 (K.-J. Pai), 107-2221-E-141-002 (R.-S. Chang) and 107-2221-E-141-001-MY3 (J.-M. Chang), from the Ministry of Science and Technology, Taiwan.

References

1. Araki, T.: Dirac's condition for completely independent spanning trees. *J. Graph Theor.* **77**, 171–179 (2014)
2. Chang, H.-Y., Wang, H.-L., Yang, J.-S., Chang, J.-M.: A note on the degree condition of completely independent spanning trees. *IEICE Trans. Fundam.* **E98-A**, 2191–2193 (2015)
3. Chang, J.-M., Pai, K.-J., Yang, J.-S., Chan, H.-C.: Embedding two disjoint multi-dimensional meshes into locally twisted cubes. *J. Internet Tech.* **16**, 541–546 (2015)
4. Chang, J.-M., Chang, H.-Y., Wang, H.-L., Pai, K.-J., Yang, J.-S.: Completely independent spanning trees on 4-regular chordal rings. *IEICE Trans. Fundam.* **E100-A**, 1932–1935 (2017)
5. Chang, N.-W., Hsieh, S.-Y.: $\{2, 3\}$ -extraconnectivities of hypercube-like networks. *J. Comput. Syst. Sci.* **79**, 669–688 (2013)
6. Chang, Y.-H., Yang, J.-S., Hsieh, S.-Y., Chang, J.-M., Wang, Y.-L.: Construction independent spanning trees on locally twisted cubes in parallel. *J. Comb. Optim.* **33**, 956–967 (2017)
7. Cheng, B., Wang, D., Fan, J.: Constructing completely independent spanning trees in crossed cubes. *Discrete Appl. Math.* **219**, 100–109 (2017)
8. Darties, B., Gastineau, N., Togni, O.: Completely independent spanning trees in some regular graphs. *Discrete Appl. Math.* **217**, 163–174 (2017)
9. Fan, G., Hong, Y., Liu, Q.: Ore's condition for completely independent spanning trees. *Discrete Appl. Math.* **177**, 95–100 (2014)
10. Guo, L., Su, G., Lin, W., Chen, J.: Fault tolerance of locally twisted cubes. *Appl. Math. Comput.* **334**, 401–406 (2018)
11. Han, Y., Fan, J., Zhang, S., Yang, J., Qian, P.: Embedding meshes into locally twisted cubes. *Inform. Sci.* **180**, 3794–3805 (2010)
12. Hasunuma, T.: Completely independent spanning trees in the underlying graph of a line digraph. *Discrete Math.* **234**, 149–157 (2001)
13. Hasunuma, T.: Completely independent spanning trees in maximal planar graphs. In: Goos, G., Hartmanis, J., van Leeuwen, J., Kučera, L. (eds.) *WG 2002*. LNCS, vol. 2573, pp. 235–245. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-36379-3_21
14. Hasunuma, T.: Minimum degree conditions and optimal graphs for completely independent spanning trees. In: Lipták, Z., Smyth, W.F. (eds.) *IWOCA 2015*. LNCS, vol. 9538, pp. 260–273. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-29516-9_22

15. Hasunuma, T., Morisaka, C.: Completely independent spanning trees in torus networks. *Networks* **60**, 59–69 (2012)
16. Hsieh, S.-Y., Huang, H.-W., Lee, C.-W.: $\{2, 3\}$ -restricted connectivity of locally twisted cubes. *Theor. Comput. Sci.* **615**, 78–90 (2016)
17. Hsieh, S.-Y., Tu, C.-J.: Constructing edge-disjoint spanning trees in locally twisted cubes. *Theor. Comput. Sci.* **410**, 926–932 (2009)
18. Hsieh, S.-Y., Wu, C.-Y.: Edge-fault-tolerant Hamiltonicity of locally twisted cubes under conditional edge faults. *J. Combin. Optim.* **19**, 16–30 (2010)
19. Hong, X., Liu, Q.: Degree condition for completely independent spanning trees. *Inform. Process. Lett.* **116**, 644–648 (2016)
20. Lai, C.-J., Chen, J.-C., Tsai, C.-H.: A systematic approach for embedding of Hamiltonian cycles through a prescribed edge in locally twisted cubes. *Inform. Sci.* **289**, 1–7 (2014)
21. Li, T.-K., Lai, C.-J., Tsai, C.-H.: A novel algorithm to embed a multi-dimensional torus into a locally twisted cube. *Theor. Comput. Sci.* **412**, 2418–2424 (2011)
22. Lin, J.-C., Yang, J.-S., Hsu, C.-C., Chang, J.-M.: Independent spanning trees vs. edge-disjoint spanning trees in locally twisted cubes. *Inform. Process. Lett.* **110**, 414–419 (2010)
23. Liu, Y.-J., Lan, J.K., Chou, W.Y., Chen, C.: Constructing independent spanning trees for locally twisted cubes. *Theor. Comput. Sci.* **412**, 2237–2252 (2011)
24. Liu, Z., Fan, J., Zhou, J., Cheng, B., Jia, Z.: Fault-tolerant embedding of complete binary trees in locally twisted cubes. *J. Parallel Distrib. Comput.* **101**, 69–78 (2017)
25. Ma, M.-J., Xu, J.-M.: Panconnectivity of locally twisted cubes. *Appl. Math. Lett.* **19**, 673–677 (2006)
26. Matsushita, M., Otachi, Y., Araki, T.: Completely independent spanning trees in (partial) k -trees. *Discuss. Math. Graph Theor.* **5**, 427–437 (2015)
27. Moinet, A., Darties, B., Gastineau, N., Baril, J.-L., Togni, O.: Completely independent spanning trees for enhancing the robustness in ad-hoc Networks. In: Proceedings of 10th IEEE International Workshop on Selected Topics in Mobile and Wireless Computing (STWiWob 2017), Rome, Italy, 9–11 October, pp. 63–70 (2017)
28. Pai, K.-J., Chang, J.-M.: Constructing two completely independent spanning trees in hypercube-variant networks. *Theor. Comput. Sci.* **652**, 28–37 (2016)
29. Pai, K.-J., Chang, J.-M.: Improving the diameters of completely independent spanning trees in locally twisted cubes. *Inform. Process. Lett.* **141**, 22–24 (2019)
30. Pai, K.-J., Tang, S.-M., Chang, J.-M., Yang, J.-S.: Completely independent spanning trees on complete graphs, complete bipartite graphs and complete tripartite graphs. In: Chang, R.S., Jain, L., Peng, S.L. (eds.) *Advances in Intelligent Systems and Applications - Volume 1. Smart Innovation, Systems and Technologies*, vol. 20, pp. 107–113. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-35452-6_13
31. Pai, K.-J., Yang, J.-S., Yao, S.-C., Tang, S.-M., Chang, J.-M.: Completely independent spanning trees on some interconnection networks. *IEICE Trans. Inform. Syst.* **E97-D**, 2514–2517 (2014)
32. Péterfalvi, F.: Two counterexamples on completely independent spanning trees. *Discrete Math.* **312**, 808–810 (2012)
33. Ren, Y., Wang, S.: The g -good-neighbor diagnosability of locally twisted cubes. *Theor. Comput. Sci.* **697**, 91–97 (2017)
34. Wei, Y.-L., Xu, M.: The g -good-neighbor conditional diagnosability of locally twisted cubes. *J. Oper. Res. Soc. China* **6**, 333–347 (2018)
35. Xu, X., Huang, Y., Zhang, P., Zhang, S.: Fault-tolerant vertex-pancyclicity of locally twisted cubes. *J. Parallel Distrib. Comput.* **88**, 57–62 (2016)

36. Xu, X., Zhai, W., Xu, J.-M., Deng, A., Yang, Y.: Fault-tolerant edge-pancyclicity of locally twisted cubes. *Inform. Sci.* **181**, 2268–2277 (2011)
37. Yang, H., Yang, X.: A fast diagnosis algorithm for locally twisted cube multiprocessor systems under the MM* model. *Comput. Math. Appl.* **53**, 918–926 (2007)
38. Yang, X., Evans, D.J., Megson, G.M.: The locally twisted cubes. *Int. J. Comput. Math.* **82**, 401–413 (2005)



Read-Once Resolutions in Horn Formulas

Hans Kleine Büning¹, P. Wojciechowski², and K. Subramani²(✉)

¹ Computer Science Institute,
University of Paderborn, Paderborn, Germany
kbcs1@uni-paderborn.de

² LDCSEE, West Virginia University,
Morgantown, WV, USA
pwojciec@mix.wvu.edu, k.subramani@mail.wvu.edu

Abstract. In this paper, we discuss the computational complexity of Read-once resolution (ROR) with respect to Horn formulas. Recall that a Horn formula is a boolean formula in conjunctive normal form (CNF), such that each clause has at most one positive literal. Horn formulas find applications in a number of domains such as program verification and logic programming. It is well-known that deduction in ProLog is based on *unification*, which in turn is based on resolution and *instantiation*. Resolution is a sound and complete procedure to check whether a boolean formula in CNF is satisfiable. Although inefficient in general, resolution has been used widely in theorem provers, on account of its simplicity and ease of implementation. This paper focuses on two variants of resolution, viz., Read-once resolution and Unit Read-once resolution (UROR). Both these variants are **sound**, but **incomplete**. In this paper, the goal is to check for the existence of proofs (refutations) of Horn formulas under these variants. We also discuss the computational complexity of determining optimal length proofs where appropriate.

1 Introduction

This paper is concerned with analyzing Horn formulas from the perspectives of Read-once resolution (ROR) and Unit Read-once resolution (UROR). A Horn clause is a clause with at most one positive literal. A Horn formula is a conjunction of Horn clauses. These formulas find applications in a number of domains, including but not limited to program verification [8, 11], logic programming [4], abstract interpretation [3] and econometrics [1]. Resolution is a sound and complete proof system introduced by Robinson [10] for checking the satisfiability of boolean formulas in conjunctive normal form (CNF). As an algorithmic strategy, resolution is not particularly efficient. However, it continues to be used

The first author was supported in part by the National Science Foundation through Award CCF-1305054.

This research was made possible by NASA WV EPSCoR Grant # NNX15AK74A.

The second author was supported in part by the Air Force Research Laboratory under US Air Force contract 88ABW-2017-1077.

© Springer Nature Switzerland AG 2019

Y. Chen et al. (Eds.): FAW 2019, LNCS 11458, pp. 100–110, 2019.

https://doi.org/10.1007/978-3-030-18126-0_9

extensively in theorem provers on account of its simplicity and ease of implementation. A resolution proof system typically declares a CNF system to be feasible, or (alternatively) finds a sequence of resolution steps which lead to the derivation of a contradiction. Such a sequence of resolution steps is called a *refutation* since it acts as a certificate, which certifies the infeasibility of the given clausal system. There are a number of variants of resolution. Each variant puts limitations on the type of resolution step that is permitted or how the resolution steps are to be counted. Three variants which are particularly important from the program verification perspective are tree-like resolution, dag-like resolution and read-once resolution. In tree-like resolution refutations input clauses can be used in multiple resolution steps, however each derived clause can only be used once. If a resolution step requires the reuse of a derived clause then that clause needs to be re-derived. Note that this re-derivation increases the length of the refutation. In dag-like resolution refutations both input and derived clauses can be used in multiple resolution steps. Finally, in read-once resolution refutations each input and derived clause can be used by only one resolution step. Note that clauses can be re-derived, and thus reused, if they can be derived from a *different* set of input clauses. A complete exposition of different types of resolutions can be found in [9] and [5].

Although the resolution proof system is both sound and complete (in general), several of its variants are not. For instance, read-once resolution (ROR) is **incomplete**, in that there exist unsatisfiable CNF formulas, which do not have read-once resolutions.

Example 1. Consider the following 2CNF formula:

$$(x_1 \vee x_2) \quad (x_3 \vee x_4) \quad (\neg x_1 \vee \neg x_3) \\ (\neg x_1 \vee \neg x_4) \quad (\neg x_2 \vee \neg x_3) \quad (\neg x_2 \vee \neg x_4)$$

We now show that this formula does not have a read-once refutation.

To derive (x_1) we need to derive $(\neg x_2)$. Similarly, to derive (x_2) we need to derive $(\neg x_1)$. However, the derivations of both $(\neg x_1)$ and $(\neg x_2)$ require the use of the clause $(x_3 \vee x_4)$.

To derive (x_3) we need to derive $(\neg x_4)$. Similarly, to derive (x_4) we need to derive $(\neg x_3)$. However, the derivations of both $(\neg x_3)$ and $(\neg x_4)$ require the use of the clause $(x_1 \vee x_2)$.

Another variant of resolution is *unit resolution*. In unit resolution, for each resolution step one of the two parent clauses must be a unit clause, i.e. a clause of the form (x_i) or $(\neg x_i)$. Although unit resolution is incomplete in general, it has been shown to be complete for Horn formulas [4]. In this paper, we study the effect of combining read-once resolution with unit resolution with respect to Horn formulas. We refer to this resolution system as Unit read-once resolution (UROR).

A natural question to ask is: How many times should a given clause be copied so that a read-once refutation can be extracted? This question was first investigated by Iwama and Miyano in [6] and leads naturally to the notion of

copy complexity (see Sect. 2). This paper investigates the copy complexity of Horn formulas with respect to unit resolution.

The principal contributions of this paper are as follows:

1. A proof that the problem of finding the shortest read-once refutation for a Horn formula is **NP-hard** (Sect. 3).
2. An algorithm that can determine if a Horn formula with m clauses and at most two literals per clause has a read-once unit resolution refutation in $O(m^2)$ time (Sect. 4).
3. Showing that the copy complexity of Horn formulas with respect to read-once unit resolution is 2^{n-1} where n is the number of variables (Sect. 5).

The rest of the paper is organized as follows. Section 2 details the problems under consideration. In Sect. 3 we cover the optimal length ROR problem for Horn formulas. Section 4 describes our work on the UROR problem for 2-Horn formulas. In Sect. 5, we examine the copy complexity of Horn formulas with respect to read-once unit resolution. Finally, Sect. 6 summarizes our results.

2 Statement of Problems

In this section, we briefly discuss the terms used in this paper. We assume that the reader is familiar with elementary propositional logic.

Definition 1. A **literal** is a variable x or its complement $\neg x$. x is called a *positive literal* and $\neg x$ is called a *negative literal*.

Definition 2. A **CNF clause** is a disjunction of literals. The empty clause, which is always false, is denoted as \sqcup .

Definition 3. A **k -CNF clause** is a CNF clause with at most k literals.

Definition 4. A **Horn clause** is a CNF clause which contains at most one positive literal.

For a single resolution step with parent clauses $(\alpha \vee x)$ and $(\neg x \vee \beta)$ and with resolvent $(\alpha \vee \beta)$, we write

$$(\alpha \vee x), (\neg x \vee \beta) \stackrel{1}{\text{RES}} (\alpha \vee \beta).$$

The variable x is called the matching or resolution variable. If for initial clauses $\alpha_1, \dots, \alpha_n$, a clause π can be generated by a sequence of resolution steps we write

$$\alpha_1, \dots, \alpha_n \stackrel{\cdot}{\text{RES}} \pi.$$

If a resolution step involves a unit clause, a clause of the form (x) or $(\neg x)$, then it is called a unit resolution step. If a resolution refutation consists of only unit resolution steps, then it is called a unit resolution refutation.

We now formally define the types of resolution refutation discussed in this paper.

Definition 5. A **Dag-like** resolution refutation is a refutation in which each clause, π , can be used in any number of resolution steps. This applies to clauses present in the original formula and those derived as a result of previous resolution steps.

Note that Dag-like refutations are unrestricted and thus are equivalent to general resolution. We now introduce several restricted forms of resolution.

Definition 6. A **Tree-like** resolution refutation is a refutation in which each derived clause, π , can be used in only one resolution step. However, clauses in the original formula can be reused and clauses can be re-derived if necessary.

Definition 7. A **Read-Once** resolution refutation is a refutation in which each clause, π , can be used in only one resolution step. This applies to clauses present in the original formula and those derived as a result of previous resolution steps.

In a read-once refutation, a clause can be reused if can be re-derived from a set of unused input clauses.

More formally, a resolution derivation $\Phi \mid_{RES} \pi$ is a read-once resolution derivation, if for all resolution steps $\pi_1 \wedge \pi_2 \mid_{RES} \pi$, we delete one instance of the clauses π_1 and π_2 from, and add a copy of the resolvent π to, the current multi-set of clauses. In other words, if U is the current multi-set of clauses, we obtain $U = (U \setminus \{\pi_1, \pi_2\}) \cup \{\pi\}$.

It is important to note Read-Once resolution is an **incomplete** refutation procedure.

We can similarly define read-once unit resolution.

Definition 8. A **Read-Once Unit** resolution refutation is a unit resolution refutation in which each clause, π , can be used in only one unit resolution step. This applies to clauses present in the original formula and those derived as a result of previous unit resolution steps.

This lets us define the concept of copy complexity with respect to read-once unit resolution.

Definition 9. A CNF formula Φ has **copy complexity** at most k , with respect to unit resolution, if there exists a multi-set of CNF clauses, Φ' such that:

1. Every clause in Φ appears at most k times in Φ' .
2. Every clause in Φ' appears in Φ .
3. Φ' has a read-once unit resolution refutation.

Let k -UROR denote the set of CNF formulas with copy complexity k with respect to unit resolution.

For any type of resolution refutation, we can define the length of that refutation in terms of the number of resolution steps.

Definition 10. The length of a resolution refutation R , is the number of resolution steps in R .

We now define the problems under consideration.

The Read-once Refutation (ROR) problem: Given a CNF formula Φ , determine whether or not Φ has a read-once refutation.

The Optimal Length Read-once Refutation (OLROR) problem: Given a CNF formula Φ , produce the read-once refutation of Φ that has minimum length.

The Read-once Unit Resolution Refutation (UROR) problem: Given a CNF formula Φ , determine whether or not Φ has a read-once unit resolution refutation.

The Copy Complexity problem: Given a CNF formula Φ , determine the copy complexity of Φ with respect to read-once unit resolution refutation.

3 The OLROR Problem for Horn Formulas

In this section, we discuss the problem of finding the optimal read-once refutation of a Horn formula.

Let Φ be an unsatisfiable Horn formula. We know that Φ has a read-once refutation [12]. The question whether Φ has a read-once resolution of length less than k is equivalent to the question whether Φ contains a minimal unsatisfiable formula consisting of at most k clauses.

Theorem 1. *Let R denote an OLROR of an unsatisfiable Horn formula Φ and let $\Phi_R \subseteq \Phi$ be the set of clauses used by R . R is also an optimal tree-like refutation of Φ and an optimal dag-like refutation of Φ . Additionally, Φ_R is a minimum unsatisfiable subset of Φ .*

Proof. Since R is a read-once refutation of Φ , R is also a tree-like and a dag-like refutation of Φ .

Assume that T is a tree like refutation of Φ such that $|T| < |R|$. Let $\Phi_T \subseteq \Phi$ be the set of clauses used by T . It follows that $|\Phi_T| \leq |T| + 1 < |R| + 1$.

Φ_T must have a read-once refutation, R_T [12]. However, $|R_T| \leq |\Phi_T| - 1 < |R|$. This contradicts the fact that R is the *optimal* read-once refutation of Φ . Thus, R is also an optimal tree-like refutation of Φ . Similarly, R is an optimal dag-like refutation of Φ .

Let $\Phi' \subseteq \Phi$ be an unsatisfiable Horn formula such that $|\Phi'| < |\Phi_R|$. Φ' must have a read-once refutation, R' . However, $|R'| \leq |\Phi'| - 1 < |\Phi_R| - 1 = |R|$. This contradicts the fact that R is the *optimal* read-once refutation of Φ . Thus, Φ_R is a minimum unsatisfiable subset of Φ . \square

We conclude that, for unsatisfiable Horn formulas, the length of the shortest resolution refutation equals the length of the shortest read-once refutation and the shortest tree-like resolution. Moreover, if k is the number of clauses of a minimal unsatisfiable subformula, then the shortest resolution refutation uses $(k - 1)$ resolution steps.

The following result is a corollary of Theorem 5.2 in [7]. However, it is included here for completeness.

Theorem 2. *The problem of deciding whether a Horn formula contains an unsatisfiable sub-formula with at most k clauses is **NP-complete**.*

Proof. We show this by a reduction from Vertex Cover. Let $G = (V, E)$ be an undirected graph where $V = \{v_1, \dots, v_n\}$ and $E = \{e_1, \dots, e_m\}$. We associate with G the Horn formula:

$$(v_1) \wedge \dots \wedge (v_n) \wedge (\neg e_1 \vee \dots \vee \neg e_m) \wedge \bigwedge_{1 \leq i \leq m, e_i = (v_{i_1}, v_{i_2})} ((\neg v_{i_1} \vee e_i) \wedge (\neg v_{i_2} \vee e_i))$$

Then there exists a subset $V' \subseteq V$ such that $|V'| \leq r$ and $V' \cap e_i$ is non-empty for every $1 \leq i \leq m$ if and only if the associated formula contains an unsatisfiable sub-formula with at most $(1 + m + r)$ clauses. (the negative clause, the clause $(\neg v \vee e_i)$ for each $1 \leq i \leq m$, and r unit clauses).

Since the problem of finding a vertex cover of size at most r is **NP-complete**, the problem of finding a read-once resolution refutation of length at most $k = (1 + m + r)$ is **NP-complete** for Horn formulas. \square

4 The UROR Problem for 2-Horn Formulas

In this section, we look at the UROR problem for 2-Horn formulas. In particular, we show that checking whether a 2-Horn formula has a unit read-once refutation is in **P**.

As in the case of Horn formulas, not every unsatisfiable 2-Horn formula has a read-once unit resolution refutation.

Example 2. Consider the formula

$$(x_1) \wedge (\neg x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2).$$

This formula is an unsatisfiable Horn formula. However, the clause (x_1) needs to be used twice in any unit resolution refutation. Thus, this formula does not have a read-once unit resolution refutation.

Let Φ be a 2-Horn formula with m clauses over n variables. We can divide Φ into a set of positive unit clauses (PU), a set of implications (ϕ), a set of negative unit clauses (NU), and a set of clauses with two negative literals (N).

We write $\Phi = PU \cup \phi \cup NU \cup N$. Note that any part (PU , ϕ , NU , or N) can be empty.

For resolution or restricted resolution calculi we say a formula Φ is *minimal* with respect to the calculus, if there exists a refutation for Φ and no proper sub-formula of Φ has a refutation.

Let \mathbf{R} be a read-once unit resolution of Φ . Let $\Phi_k = PU_k \cup \phi_k \cup NU_k \cup N_k$ be the set of clauses derived from Φ after k steps of \mathbf{R} . Let us consider the $(k+1)^{th}$ resolution step of \mathbf{R} .

Note that this resolution step must resolve one of the following clause pairs:

1. A clause of the form $(x_i) \in PU_k$ and a clause of the form $(\neg x_i) \in NU_k$. In this case, this resolution step produces the empty cause. Note that this is the last step of the refutation.
2. A clause of the form $(x_i) \in PU_k$ and a clause of the form $(\neg x_i \vee \neg x_j) \in N_k$. In this case, the resolution produces the negative unit clause $(\neg x_j)$. Thus, $PU_{k+1} = PU_k \setminus \{(x_i)\}$, $N_{k+1} = N_k \setminus \{(\neg x_i \vee \neg x_j)\}$, and $NU_{k+1} = NU_k \cup \{(\neg x_j)\}$. Note that $|PU_{k+1} \cup NU_{k+1}| = |PU_k \cup NU_k|$ and $|NU_{k+1} \cup N_{k+1}| = |NU_k \cup N_k|$.
3. A clause of the form $(x_i) \in PU_k$ and a clause of the form $(\neg x_i \vee x_j) \in \phi_k$. In this case, the resolution produces the positive unit clause (x_j) . Thus, $PU_{k+1} = (PU_k \setminus \{(x_i)\}) \cup \{(x_j)\}$, $N_{k+1} = N_k$, and $NU_{k+1} = NU_k$. Note that $|PU_{k+1} \cup NU_{k+1}| = |PU_k \cup NU_k|$ and $|NU_{k+1} \cup N_{k+1}| = |NU_k \cup N_k|$.
4. A clause of the form $(\neg x_i) \in NU_k$ and a clause of the form $(x_i \vee \neg x_j) \in \phi_k$. In this case, the resolution produces the negative unit clause $(\neg x_j)$. Thus, $PU_{k+1} = PU_k$, $N_{k+1} = N_k$, and $NU_{k+1} = (NU_k \setminus \{(\neg x_i)\}) \cup \{(\neg x_j)\}$. Note that $|PU_{k+1} \cup NU_{k+1}| = |PU_k \cup NU_k|$ and $|NU_{k+1} \cup N_{k+1}| = |NU_k \cup N_k|$.

Let $\Phi = PU \cup \phi \cup NU \cup N$ be minimal with respect to read-once unit resolution. Let \mathbf{R} be a read-once unit resolution refutation of Φ and let $(x_1) \wedge (\neg x_1) \vdash_{RES}^1 \perp$, be the last resolution step of \mathbf{R} . Based on the preceding arguments, Φ must have the following properties:

1. $|NU \cup N| = 1$. Note that every prior resolution step of \mathbf{R} preserves the size of $NU \cup N$. Before the last resolution step of \mathbf{R} , $NU \cup N = \{(\neg x_i)\}$. Thus, before the last resolution step $|NU \cup N| = 1$. This means that this must have been true for the original formula.
2. $|PU \cup NU| = 2$. Note that every prior resolution step of \mathbf{R} preserves the size of $PU \cup NU$. Before the last resolution step of \mathbf{R} , $PU \cup NU = \{(x_i), (\neg x_i)\}$. Thus, before the last resolution step $|PU \cup NU| = 2$. This means that this must have been true for the original formula.

Thus, Φ must satisfy one of the following conditions:

1. $|PU| = 1$, $|NU| = 1$, and $|N| = 0$.
2. $|PU| = 2$, $|NU| = 0$, and $|N| = 1$.

This means that we only need to consider two types of read-once unit resolution refutations.

1. Refutations which use a single positive unit clause from PU and a single negative unit clause from NU . We refer to this as a Type 1 refutation.
2. Refutations which use two positive unit clauses from PU and a single negative clause from N . We refer to this as a Type 2 refutation.

From an arbitrary 2-Horn formula $\Phi = PU \cup \phi \cup NU \cup N$, we can create a directed graph $\mathbf{G} = \langle \mathbf{V}, \mathbf{E} \rangle$ as follows:

1. For each variable x_i in Φ , add the vertex v_i to \mathbf{V} .
2. For each clause $(x_i \vee \neg x_j)$, add the edge (v_j, v_i) to \mathbf{E} .
3. Let $S = \{v_i | (x_i) \in PU\}$ and $T = \{v_j | (\neg x_j) \in NU\}$.

We utilize \mathbf{G} to find read-once unit resolution refutations.

Lemma 1. *The clause (x_i) is derivable from Φ by a read-once unit resolution if and only if v_i is reachable in \mathbf{G} from a vertex in S .*

Proof. Assume that (x_i) is derivable from Φ by a read-once unit resolution \mathbf{R} . We will show that v_i is reachable from a vertex in S by induction based on the number of resolution steps in \mathbf{R} .

If \mathbf{R} has no resolution steps then $(x_i) \in \Phi$. Thus, $(x_i) \in PU$ and, by construction, $v_i \in S$. Thus, v_i is trivially reachable from a vertex in S .

Now suppose that this holds true of all resolutions with k steps. If \mathbf{R} has $(k+1)$ steps, then the last resolution step of \mathbf{R} must be $(x_j) \wedge (\neg x_j \vee x_i) \stackrel{1}{\text{RES}} (x_i)$ for some clause (x_j) . This means that (x_j) must be derivable from Φ by a read-once unit resolution with k steps. Thus, by the induction hypothesis, v_j is reachable from a vertex in S . Thus, $(\neg x_j \vee x_i) \in \Phi$ since non-unit clauses are not derivable from 2-Horn clauses by unit resolution. This means that $(\neg x_j \vee x_i) \in \phi$ and, by construction, the edge (v_j, v_i) is in \mathbf{E} . Thus, v_i is also reachable from a vertex in S .

Now assume that v_i is reachable from a vertex $v_j \in S$. Thus, there must be a simple path p in \mathbf{G} from v_j to v_i . Let the edges in p be $(v_j, v_{p_1}), (v_{p_1}, v_{p_2}), \dots, (v_{p_k}, v_i)$. By construction of \mathbf{G} , the clauses $(x_j), (\neg x_j \vee x_{p_1}), (\neg x_{p_1} \vee x_{p_2}), \dots, (\neg x_{p_k} \vee x_i)$ are all in Φ .

It is easy to see that these clauses can be used to derive (x_i) through unit resolution. Since p was simple no clause is used more than once. Thus, this is a read-once unit resolution of (x_i) . \square

Theorem 3. *Whether the formula Φ has a Type 1 refutation can be determined in linear time.*

Proof. Φ has a Type 1 refutation if and only if for some clause $(\neg x_t) \in \Phi$, we can derive the clause (x_t) from Φ by a read-once unit resolution \mathbf{R} . Thus, by Lemma 1, some vertex $v_t \in T$ must be reachable from a vertex in S . This can be checked by doing a depth-first search in \mathbf{G} . \square

Theorem 4. *Whether the formula Φ has a Type 2 refutation can be determined in $O(m^2)$ time.*

Proof. Φ has a Type 2 refutation if and only if for some clause $(\neg x_{t_1} \vee \neg x_{t_2}) \in \Phi$, we can derive the clauses (x_{t_1}) and (x_{t_2}) from Φ by read-once unit resolution \mathbf{R} . Thus, by Lemma 1, both v_{t_1} and v_{t_2} must be reachable from vertices in S . Thus, there is a path p_1 from a vertex in S to v_{t_1} and there is a path p_2 from a vertex in S to v_{t_2} . Since \mathbf{R} is a read-once unit resolution, p_1 and p_2 cannot share any edges and cannot have the same starting vertex.

If we are given vertices v_{t_1} and v_{t_2} , then we need to check if \mathbf{G} has such a pair of edge-disjoint paths.

From \mathbf{G} , S , v_{t_1} , and v_{t_2} , we can construct flow network \mathbf{G}' as follows:

1. For each vertex v_i in \mathbf{G} , add the node v_i to \mathbf{G}' .
2. For each edge (v_i, v_j) in \mathbf{G} add the edge (v_i, v_j) to \mathbf{G}' with capacity 1.
3. Add a source s and the edges (s, v_i) for $v_i \in S$ to \mathbf{G}' .
4. Add a sink t and the edges (v_{t_1}, t) , and (v_{t_2}, t) to \mathbf{G}' .

The desired paths exist if and only if a flow of 2 can be pushed from s to t in \mathbf{G}' .

Thus, Φ has a Type 2 refutation if and only if, for some clause $(\neg x_{t_1} \vee \neg x_{t_2}) \in \Phi$, the graph \mathbf{G}' has a max flow of 2.

Note that all edges in \mathbf{G}' have capacity 1, and that t has only two incoming edges. This means that the maximum flow is at most 2. Thus, we can find the max flow from s to t in \mathbf{G}' in $O(m)$ time where m is the number of clauses in Φ [2]. Since we need to find this flow for each clause of the form $(\neg x_{t_1} \vee \neg x_{t_2})$, determining if Φ has a Type 2 refutation can be accomplished in $O(m^2)$ time. \square

Thus, checking if Φ has a read-once unit resolution refutation can be accomplished by checking to see if it has either a Type 1 refutation or a Type 2 refutation. This can be accomplished in $O(m^2)$ time since checking for Type 2 refutations is the more time consuming process.

5 UROR Copy Complexity of Horn Formulas

We now examine the copy complexity of Horn formulas with respect to unit resolution.

Theorem 5. *The copy complexity of Horn formulas with respect to unit resolution is at most 2^{n-1} where n is the number of variables.*

Proof. Suppose Φ is an unsatisfiable Horn formula. Note that adding clauses to a system cannot increase the copy complexity. Thus, we can assume without loss of generality that Φ is minimal unsatisfiable. Let $CC(n)$ denote the copy complexity of a minimal unsatisfiable Horn formula with n variables. We will show that $CC(n) \leq 2^{n-1}$. For a clause ϕ_j of Φ let $N_c(\phi_j)$ be the number of copies of ϕ_j needed for a read-once unit resolution refutation.

Let Φ be a minimal unsatisfiable Horn formula with n variables. Thus, Φ has $(n + 1)$ clauses. If $n = 1$, then Φ has the form $(x) \wedge (\neg x)$. This formula has

a read-once unit resolution refutation. Thus $CC(1) = 1 \leq 2^0$. Also note that $\sum_{\phi_j \in \Phi} N_c(\phi_j) = 2 \leq 2^1$.

Now assume that $CC(k) \leq 2^{k-1}$, and that for each minimal unsatisfiable formula Φ' with k variables, $\sum_{\phi'_j \in \Phi'} N_c(\phi'_j) \leq 2^k$. If $n = k + 1$, then Φ has the form $(x) \wedge (\neg x \vee \alpha_1) \wedge \dots \wedge (\neg x \vee \alpha_t) \wedge \sigma_{t+1} \wedge \dots \wedge \sigma_{k+1}$. A read-once unit resolution refutation needs to use the clause (x) to eliminate each instance of $\neg x$. Thus, we need a copy of the clause (x) for each copy of $(\neg x \vee \alpha_i)$ for $i = 1 \dots t$. Let Φ' be the formula $\alpha_1 \wedge \dots \wedge \alpha_t \wedge \sigma_{t+1} \wedge \dots \wedge \sigma_{k+1}$. Note that Φ' is a minimal unsatisfiable formula with $n - 1 = k$ variables. Thus,

$$\sum_{j=1}^t N_c(\alpha_i) \leq \sum_{\phi'_j \in \Phi'} N_c(\phi'_j) \leq 2^k.$$

This means that we need at most 2^k copies of the clause (x) . Thus, $CC(k+1) \leq 2^k$ and $\sum_{\phi_j \in \Phi} N_c(\phi_j) \leq 2^k + 2^k = 2^{k+1}$. \square

Theorem 6. *There exists a Horn formula with copy complexity 2^{n-1} where n is the number of variables.*

Proof. Consider the following clauses:

$$\begin{array}{ccc} (\neg x_1 \vee \neg x_2 \vee \dots \vee \neg x_n) & (x_1 \vee \neg x_2 \vee \dots \vee \neg x_n) & (x_2 \vee \neg x_3 \vee \dots \vee \neg x_n) \\ \dots & (x_{n-1} \vee \neg x_n) & (x_n) \end{array}$$

Let us consider the clause $(\neg x_1 \vee \neg x_2 \vee \dots \vee \neg x_n)$. To eliminate this clause through unit resolution, we need to derive the clauses $(x_1), (x_2), \dots, (x_n)$. We will prove by induction, that for each $i = 1 \dots n$, 2^{i-1} copies of the clause $(x_i \vee \neg x_{i+1} \vee \dots \vee \neg x_n)$ are required.

To eliminate $\neg x_1$ from the clause we only need one copy of the clause (x_1) , thus we only need $1 = 2^0$ copies of the clause $(x_1 \vee \neg x_2 \vee \dots \vee \neg x_n)$.

Now assume that for each $i < k$, we need to use 2^{i-1} copies of the clause $(x_i \vee \neg x_{i+1} \vee \dots \vee \neg x_n)$. Each of these clauses uses the literal $\neg x_k$, as does the clause $(\neg x_1 \vee \neg x_2 \vee \dots \vee \neg x_n)$. Thus, we need a total of $1 + \sum_{i=1}^{k-1} 2^{i-1} = 2^{k-1}$ copies of the clause (x_k) to cancel every instance of $\neg x_k$. The only clause with the literal x_k is the clause $(x_k \vee \neg x_{k+1} \vee \dots \vee \neg x_n)$. Thus, we need to use 2^{k-1} copies of this clause, as desired.

This means that we need a total of 2^{n-1} copies of the clause (x_n) . Thus, the copy complexity of this formula is 2^{n-1} with respect to unit resolution. \square

From these two results, it is easy to see that the copy complexity of Horn formulas with respect to read-once unit resolution refutation is 2^{n-1} .

6 Conclusion

In this paper, we studied two proof systems, viz., Read-once resolution (ROR) and Unit read-once resolution (UROR) from the perspective of Horn clause systems. Our work is motivated by two important factors, viz., the ubiquitousness

of the resolution proof system in SMT solvers and the wide applicability of Horn formulas. As discussed before, the ROR and UROR proof systems are incomplete for general CNF formulas, although ROR is complete for Horn formulas. Note that if an unsatisfiable formula has a read-once refutation or unit read-once refutation, then this refutation is necessarily short. This is in contrast to general resolution proofs, which could be exponentially large with respect to input size. Our investigations established that the problem of checking whether a Horn clause system has a read-once refutation of length at most k is **NP-hard**. We also showed that the problem of finding a UROR for 2-Horn formulas is in **P**. Finally, we discussed the copy complexity of Horn formulas with respect to unit resolution and obtained an exponential lower bound.

References

1. Chandrasekaran, R., Subramani, K.: A combinatorial algorithm for Horn programs. *Discrete Optim.* **10**, 85–101 (2013)
2. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: *Introduction to Algorithms*. MIT Press, Cambridge (2001)
3. Cousot, P., Cousot, R.: Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: *POPL*, pp. 238–252 (1977)
4. Dowling, W.F., Gallier, J.H.: Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *J. Log. Program.* **1**(3), 267–284 (1984)
5. Gallier, J.H.: *Logic for Computer Science: Foundations for Automatic Theorem Proving*. Dover Books on Computer Science, 1st edn. Dover Publications (2015)
6. Iwama, K., Miyano, E.: Intractability of read-once resolution. In: *Proceedings of the 10th Annual Conference on Structure in Complexity Theory (SCTC 1995)*, Los Alamitos, CA, USA, June 1995, pp. 29–36. IEEE Computer Society Press (1995)
7. Liberatore, P.: Redundancy in logic ii: 2CNF and horn propositional formulae. *Artif. Intell.* **172**(2), 265–299 (2008)
8. Owre, S., Shankar, N.: *The formal semantics of PVS*, March 1999
9. Robinson, J.A., Voronkov, A.: *Handbook of Automated Reasoning* (2 volume set). MIT Press, Cambridge (2001)
10. Robinson, J.A.: A machine-oriented logic based on the resolution principle. *J. ACM* **12**(1), 23–41 (1965)
11. Rushby, J.M., Owre, S., Shankar, N.: Subtypes for specifications: predicate subtyping in PVS. *IEEE Trans. Softw. Eng.* **24**(9), 709–720 (1998)
12. Szeider, S.: NP-completeness of refutability by literal-once resolution. In: Goré, R., Leitsch, A., Nipkow, T. (eds.) *IJCAR 2001*. LNCS, vol. 2083, pp. 168–181. Springer, Heidelberg (2001). <https://doi.org/10.1007/3-540-45744-5-13>



Vertex-Critical (P_5 , banner)-Free Graphs

Qingqiong Cai¹, Shenwei Huang¹, Tao Li¹, and Yongtang Shi^{2(✉)}

¹ College of Computer Science, Nankai University, Tianjin 300350, China

² Center for Combinatorics and LPMC, Nankai University, Tianjin 300071, China
shi@nankai.edu.cn

Abstract. Given two graphs H_1 and H_2 , a graph is (H_1, H_2) -free if it contains no induced subgraph isomorphic to H_1 or H_2 . Let P_t and C_t be the path and the cycle on t vertices, respectively. A banner is the graph obtained from a C_4 by adding a new vertex and making it adjacent to exactly one vertex of the C_4 . For a fixed integer $k \geq 1$, a graph G is said to be k -vertex-critical if the chromatic number of G is k and the removal of any vertex results in a graph with chromatic number less than k . The study of k -vertex-critical graphs for graph classes is an important topic in algorithmic graph theory because if the number of such graphs that are in a given hereditary graph class is finite, then there is a polynomial-time algorithm to decide if a graph in the class is $(k - 1)$ -colorable. In this paper, we show that there are finitely many 6-vertex-critical (P_5, banner) -free graphs. This is one of the few results on the finiteness of k -vertex-critical graphs when $k > 4$. To prove our result, we use the celebrated Strong Perfect Graph Theorem and well-known properties on k -vertex-critical graphs in a creative way.

1 Introduction

All graphs in this paper are finite and simple. Let \mathcal{H} be a set of graphs. A graph G is \mathcal{H} -free if it does not contain any member in \mathcal{H} as an induced subgraph. In case that \mathcal{H} consists of a single graph H or two graphs H_1 and H_2 , we write H -free and (H_1, H_2) -free instead of $\{H\}$ -free and $\{H_1, H_2\}$ -free, respectively. A class of graphs is said to be *hereditary* if every induced subgraph of a member in the class is also in the class. It is easy to see that a graph class is hereditary if and only if it is \mathcal{H} -free for some set \mathcal{H} of graphs. For instance, the class of bipartite graphs coincides with the class of graphs that does not contain any odd cycle as an induced subgraph.

A k -coloring of a graph G is a function $\phi : V(G) \rightarrow \{1, \dots, k\}$ such that $\phi(u) \neq \phi(v)$ whenever u and v are adjacent in G . Equivalently, a k -coloring of G can be viewed as a partition of $V(G)$ into k stable sets. We say that G is k -colorable if it admits a k -coloring. The *chromatic number* of G , denoted by $\chi(G)$, is the minimum number k such that G is k -colorable. A graph G is k -chromatic if $\chi(G) = k$. We say that G is k -critical if it is k -chromatic and $\chi(G - e) < \chi(G)$ for any edge $e \in E(G)$. For instance, K_2 is the only 2-critical graph and odd cycles are the only 3-critical graphs. A graph is *critical* if it is k -critical for some

integer $k \geq 1$. Critical graphs were first defined and studied by Dirac [7–9] in the early 1950s, and then by Gallai and Ore [11, 12, 20] among many others, and more recently by Kostochka and Yancey [19].

A weaker notion of criticality is the so-called vertex-critical graphs. A graph G is k -vertex-critical if $\chi(G) = k$ and $\chi(G - v) < k$ for any $v \in V(G)$. For a set \mathcal{H} of graphs and a graph G , we say that G is k - \mathcal{H} -vertex-critical-free if it is k -vertex-critical and \mathcal{H} -free. In this paper, we study k -vertex-critical \mathcal{H} -free graphs. We are mainly interested in the following question: *given a set \mathcal{H} of graphs and an integer $k \geq 1$, are there only finitely many k -vertex critical \mathcal{H} -free graphs?* This question is important in the study of algorithmic graph theory because of the following theorem.

Theorem 1 (Folklore). *For a given set \mathcal{H} of graphs and an integer $k \geq 1$, if the set of all k -vertex-critical \mathcal{H} -free graphs is finite, then there is a polynomial-time algorithm to determine whether an \mathcal{H} -free graph is $(k - 1)$ -colorable.*

Proof. Suppose that $\{F_1, \dots, F_r\}$ is the set of all k -vertex-critical \mathcal{H} -free graphs, where $r \geq 1$ is a constant and $|V(F_i)| = n_i$ for $1 \leq i \leq r$. Let G be an \mathcal{H} -free graph with n vertices. Observe that G is $(k - 1)$ -colorable if and only if G is F_i -free for each $1 \leq i \leq r$. We now brute-force on all n_i -tuples of $V(G)$ and check whether or not the given tuple induces a subgraph isomorphic to F_i . Since F_i has n_i vertices, there are at most n^{n_i} such tuples, and this implies that it takes $O(n^{n_i})$ time to decide if G is F_i -free. Therefore, it takes $\sum_i O(n^{n_i}) \leq O(n^{\max\{n_1, \dots, n_r\}})$ time to determine if G is $(k - 1)$ -colorable. Since r is finite, the running time is a polynomial function in n . \square

Let K_n be the complete graph on n vertices. Let P_t and C_t denote the path and the cycle on t vertices, respectively. We now review some known results in the study of k -vertex-critical \mathcal{H} -free graphs. The only k -vertex-critical perfect graph (see the definition in Sect. 2) is the complete graph K_k , and there is no 5-vertex-critical planar graphs by the Four Color Theorem. Another class of graphs that has been extensively studied recently is the class of P_t -free graphs. In [2], it was shown that there are finitely many 4-vertex-critical P_5 -free graphs. This result was later generalized to P_6 -free graphs [4]: there are 80 4-vertex-critical P_6 -free graphs. In the same paper, an infinite family of 4-vertex-critical P_7 -free graphs was constructed. Randerath and Schiermeyer [21] have shown that there are finitely many 4-vertex-critical (P_6, C_3) -free graphs. Hell and Huang [14] determined all four 4-vertex-critical (P_6, C_4) -free graphs. Goedgebeur and Schaudt [13] prove that there are finitely many 4-vertex-critical (P_t, C_4) -free graphs for $t = 7, 8$, and that there are finitely many 4-vertex-critical (P_7, C_5) -free graphs. It was also known that there are finitely many 5-vertex-critical (P_5, C_5) -free graphs [16] and (P_6, banner) -free graphs [17].

For two graphs G and H , we use $G + H$ to denote the *disjoint union* of G and H . For a positive integer r , we use rG to denote the disjoint union of r copies of G . For $s, r \geq 1$, let $K_{r,s}$ be the complete bipartite graph with one part of size r and the other part of size s . For the class of H -free graphs, it was shown in [3, 4] that the set of 4-vertex-critical H -free graphs is finite if and only if H is a

subgraph of P_6 , $2P_3$ or $P_4 + rP_1$ for some $r \geq 1$. The finiteness for every fixed $k \geq 1$ was also shown for some other classes, e.g., for $(P_5, \overline{P_5})$ -free graphs [6], (P_6, C_4) -free graphs [14], $(P_t, K_{s,r})$ -free graphs [18] for any $t, s, r \geq 1$.

Our Contributions. A *banner* is the graph obtained from a C_4 by adding a new vertex and making it adjacent to exactly one vertex of the C_4 . In this paper, we prove that there are finitely many 6-vertex-critical (P_5, banner) -free graphs. We note that most known results are on 4-critical graphs, and that the larger k is, the more difficult it is to prove finiteness of k -vertex-critical graphs. In addition, the case $k = 5$ has been done in [17]. Our result is one of the few results for finiteness of k -vertex-critical graphs when $k > 4$. Moreover, it follows from Theorem 1 that our result generalizes the previous result on the polynomial-time algorithm for 5-coloring P_5 -free graphs when restricted to banner-free graphs [15].

The remainder of the paper is organized as follows. We present some preliminaries in Sect. 2 and give structural properties around an induced C_5 in a (P_5, banner) -free graph in Sect. 3. We show that there are finitely many 6-vertex-critical (P_5, banner) -free graphs in Sect. 4. We conclude our paper in Sect. 5.

2 Preliminaries

For general graph theory notation we follow [1]. The *complement* of a graph G is denoted by \overline{G} . For $k \geq 4$, an induced cycle of length k is also called a k -hole. A k -hole is an *odd hole* (resp. *even hole*) if k is odd (resp. even). A k -antihole is the complement of a k -hole. Odd and even antiholes are defined analogously.

Let $G = (V, E)$ be a graph. The *neighborhood* of a vertex v , denoted by $N_G(v)$, is the set of neighbors of v . For a set $X \subseteq V(G)$, let $N_G(X) = \bigcup_{v \in X} N_G(v) \setminus X$ and $N_G[X] = N(X) \cup X$. The *degree* of v , denoted by $d_G(v)$, is equal to $|N_G(v)|$. We shall omit the subscript G when the context is clear. The *minimum degree* of G over all vertices in G is denoted by $\delta(G)$. For $x \in V$ and $S \subseteq V$, we denote by $N_S(x)$ the set of neighbors of x that are in S , i.e., $N_S(x) = N_G(x) \cap S$. For $X, Y \subseteq V$, we say that X is *complete* (resp. *anti-complete*) to Y if every vertex in X is adjacent (resp. non-adjacent) to every vertex in Y . A vertex subset $S \subseteq V$ is *stable* if no two vertices in S are adjacent. A *clique* is the complement of a stable set. A vertex subset $K \subseteq V$ is a *clique cutset* if $G - K$ has more connected components than G and K is a clique. A vertex is *universal* in G if it is adjacent to all other vertices in G . For $S \subseteq V$, the subgraph *induced* by S is denoted by $G[S]$. We often write S for $G[S]$ if the context is clear. We say that a vertex w *distinguishes* two vertices u and v if w is adjacent to exactly one of u and v . Two non-adjacent vertices u and v are said to be *comparable* if $N(u) \subseteq N(v)$ or $N(v) \subseteq N(u)$.

The following lemma is well-known in the study of k -vertex-critical graphs.

Lemma 1 ([22]). *Let G be a k -vertex-critical graph. Then the following holds: (i) $\delta(G) \geq k - 1$; (ii) G contains no clique cutsets; (iii) G contains no pair of comparable vertices.*

Another useful result is the following lemma.

Lemma 2 ([10]). *Let G be a connected bipartite graph. Then G is P_5 -free if and only if G is $2P_2$ -free.*

The *clique number* of G , denoted by $\omega(G)$, is the size of a largest clique in G . A graph G is *perfect* if $\chi(H) = \omega(H)$ for every induced subgraph H of G . Another result we use is the well-known Strong Perfect Graph Theorem.

Theorem 2 (The Strong Perfect Graph Theorem [5]). *A graph is perfect if and only if it does not contain any odd hole or odd antihole as an induced subgraph.*

Observe that the only k -vertex-critical perfect graphs is the complete graph K_k . This can be seen as follows. Let G be a k -vertex-critical perfect graphs. Since G is k -chromatic, it must contain K_k as an induced subgraph by the definition of perfect graphs. Since G is k -vertex-critical, it follows that G is isomorphic to K_k .

3 Structure Around a 5-Hole

Let $G = (V, E)$ be a graph and H be an induced subgraph of G . We partition $V \setminus V(H)$ into subsets with respect to H as follows: for any $X \subseteq V(H)$, we denote by $S(X)$ the set of vertices in $V \setminus V(H)$ that have X as their neighborhood among $V(H)$, i.e.,

$$S(X) = \{v \in V \setminus V(H) : N_{V(H)}(v) = X\}.$$

For $0 \leq j \leq |V(H)|$, we denote by S_j the set of vertices in $V \setminus V(H)$ that have exactly j neighbors in $V(H)$. Note that $S_j = \bigcup_{X \subseteq V(H): |X|=j} S(X)$. We say that a vertex in S_j is a *j -vertex*.

Let G be a (P_5, banner) -free graph and $C = 1, 2, 3, 4, 5$ be an induced C_5 in G . We partition $V \setminus V(C)$ with respect to C as above. All indices below are modulo five. We now prove a number of useful properties of $S(X)$ using the fact that G is (P_5, banner) -free. All properties are proved for $i = 1$ due to symmetry.

- (1) $S(i) = S(i, i + 1) = \emptyset$.
 $S(1) \cup S(1, 2)$ contains a vertex, then $x, 1, 5, 4, 3$ induces a P_5 . □
- (2) $S(i, i + 2) = S(i, i + 2, i - 2) = \emptyset$.
 If $S(1, 3) \cup S(1, 3, 4)$ contains a vertex x , then $\{1, 2, 3, x, 5\}$ induces a banner. □
- (3) $S(i - 1, i, i + 1)$ and $S(i - 1, i - 2, i + 2, i + 1)$ are cliques.
 If $S(5, 1, 2)$ contains two non-adjacent vertices x and y , then $\{x, y, 5, 2, 3\}$ induces a banner. Similarly, if $S(2, 3, 4, 5)$ contains two non-adjacent vertices x and y , then $\{x, y, 5, 1, 3\}$ induces a banner. □
- (4) S_0 is anti-complete to $S_3 \cup S_4$.
 Suppose that $a \in S_0$ is adjacent to $x \in S_3 \cup S_4$. By (2), $x \in S(i, i + 1, i + 2)$ or $x \in S(i + 1, i + 2, i + 3, i + 4)$ for some $i \in V(C)$. By symmetry, we may assume that $i = 1$. If $x \in S(1, 2, 3)$, then $a, x, 1, 5, 4$ induces a P_5 . If $x \in S(1, 2, 3, 4)$, then $\{1, x, 4, 5, a\}$ induces a banner. □

- (5) Let $x, y \in S_5$ with $xy \notin E$. Then every vertex in $S_3 \cup S_4$ is adjacent to at least one of x and y .

Assume that $t \in S_3 \cup S_4$. Since t is adjacent to at least one but not all vertices on C , there exists a vertex $i \in C$ such that t is adjacent to i but not to $i + 2$. If t is adjacent to neither x nor y , then $\{t, i, i + 2, x, y\}$ induces a banner. \square

4 The Main Result

In this section, we show that there are finitely many 6-vertex-critical (P_5, banner) -free graphs.

Theorem 3. *There are finitely many 6-vertex-critical (P_5, banner) -free graphs.*

The proof of the theorem relies on the following three lemmas.

Lemma 3. *Let G be a 6-vertex-critical (P_5, banner) -free graph. If G contains an induced $\overline{C_9}$, then G has a finite order.*

Lemma 4. *Let G be a 6-vertex-critical (P_5, banner) -free graph. If G contains an induced $\overline{C_7}$, then G has a finite order.*

Lemma 5. *Let G be a 6-vertex-critical (P_5, banner) -free graph. If G contains an induced C_5 , then G has a finite order.*

We now prove the theorem using Lemmas 3, 4 and 5 and then prove the lemmas in the following three subsections.

Proof (Proof of Theorem 3). Let G be a 6-vertex critical (P_5, banner) -free graph. It follows from Lemmas 3, 4 and 5 that if G contains an induced $\overline{C_9}$, $\overline{C_7}$ or C_5 , then the order of G is finite. Therefore, we may assume that G is also $\{\overline{C_9}, \overline{C_7}, C_5\}$ -free. Since G is 6-vertex-critical and $\chi(\overline{C_{11}}) = 6$, it follows that $G = \overline{C_{11}}$ if G contains an induced $\overline{C_{11}}$. Therefore, we may assume that G is also $\overline{C_{11}}$ -free. Since G is 6-chromatic and $\chi(\overline{C_{2t+1}}) \geq 7$ for any $t \geq 6$, it follows that G does not contain any odd antihole. Moreover, since G is P_5 -free, it does not contain any odd hole. It then follows from Theorem 2 that G is perfect. Therefore, G is isomorphic to K_6 . Hence, the theorem follows. \square

We denote by $G + ku$ the graph obtained from a graph G by adding a clique of size k and adding an edge between every vertex in G and every vertex in the new clique. Observe that $\overline{C_9} + 1u$ and $\overline{C_7} + 2u$ are 6-vertex-critical (P_5, banner) -free graphs.

4.1 Proof of Lemma 3

Proof. Let $C = 1, 2, 3, 4, 5, 6, 7, 8, 9$ be an induced $\overline{C_9}$ such that $ij \in E$ if and only if $|i - j| > 1$. All indices are modulo 9. We partition $V(G)$ with respect to C . If $S_9 \neq \emptyset$, then G contains $(\overline{C_9} + 1u)$, which is a 6-chromatic graph. Since G is 6-vertex-critical, it follows that G is isomorphic to $(\overline{C_9} + 1u)$ and the lemma holds. Therefore, we may assume that $S_9 = \emptyset$.

Claim 1. $S_i = \emptyset$ for $1 \leq i \leq 4$.

Proof (Proof of Claim 1). Let $x \in S(X)$ for some $X \subseteq C$ with $1 \leq |X| \leq 4$. We show that there is an induced $C_4 \subseteq C$ containing exactly one neighbor of x and this C_4 together with x gives an induced banner in G . Let $x \in S_k$ for some $1 \leq k \leq 4$. Since x has at most 4 neighbors on C , there exists an index i such that x is adjacent to neither $i - 4$ nor $i + 4$ (since the vertex cover number of C_9 is 5). If x has a neighbor in $\{i - 2, i - 1, i, i + 1, i + 2\}$, then there exist $j, j + 1 \in \{i - 2, i - 1, i, i + 1, i + 2\}$ such that x distinguishes j and $j + 1$. For otherwise x would be complete to $\{i - 2, i - 1, i, i + 1, i + 2\}$, which contradicts the assumption that x has at most 4 neighbors on C . Then $\{j, j + 1, i - 4, i + 4, x\}$ induces a banner. The remaining case is that x is anti-complete to $V(C) \setminus \{i - 3, i + 3\}$. By symmetry, we assume that x is adjacent to $i + 3$. Then $\{i - 2, i - 1, i + 2, i + 3, x\}$ induces a banner. This shows that $S_i = \emptyset$ for $1 \leq i \leq 4$. □

Claim 2. $S_0 = \emptyset$.

Proof (Proof of Claim 2). Suppose not. Let A be a connected component of S_0 . By Claim 2 and the connectivity of G , some vertex $a \in A$ has a neighbor $n \in S_i$ for some $5 \leq i \leq 8$. It can be readily seen that there exists an index i such that n is adjacent to $i - 4$ and $i + 4$ but not to some j where $j \in \{i - 2, i - 1, i, i + 1, i + 2\}$. Now $\{i - 4, i + 4, j, n, a\}$ induces a banner, a contradiction. This proves that $S_0 = \emptyset$. □

Claim 3. For every $X \subseteq C$ with $5 \leq |X| \leq 8$, $S(X)$ is a clique.

Proof (Proof of Claim 3). Let $X \subseteq C$ be an arbitrary set with $5 \leq |X| \leq 8$. Since $5 \leq |X| \leq 8$, there exists an index $i \in C$ such that $i, i + 1 \in X$ but $i + 2 \notin X$. If $S(X)$ contains two non-adjacent vertices x and y , then $\{i, i + 1, i + 2, x, y\}$ induces a banner. Therefore, $S(X)$ is a clique. □

Since G is 6-vertex-critical, $G = K_6$ if G contains a K_6 . Therefore, we may assume that G is K_6 -free. By Claim 3, it follows that $|S(X)| \leq 3$ for every $X \subseteq C$ with $5 \leq |X| \leq 8$. By Claims 1 and 2, $V(G) = C \cup S_5 \cup S_6 \cup S_7 \cup S_8$. Therefore, $|V(G)| \leq 9 + 3\binom{9}{5} + \binom{9}{6} + \binom{9}{7} + \binom{9}{8} = 794$. □

4.2 Proof of Lemma 4

Proof. Let $C = 1, 2, 3, 4, 5, 6, 7$ be an induced $\overline{C_7}$ such that $ij \in E$ if and only if $|i - j| > 1$. All indices are modulo 7. We partition $V(G)$ with respect to C . If S_7 is not stable, then G is isomorphic to $(\overline{C_7} + 2u)$ and so the lemma holds. Therefore, we may assume that S_7 is stable.

Claim 4. $S_i = \emptyset$ for $1 \leq i \leq 3$.

Proof (Proof of Claim 4). Let $x \in S_i$ for some $1 \leq i \leq 3$. Since x has at most three neighbors on C , there exists a vertex $i \in V(C)$ such that x is adjacent to neither $i - 3$ nor $i + 3$ and x is adjacent to some vertex in $\{i - 1, i, i + 1\}$. If x is complete to $\{i - 1, i, i + 1\}$, then $\{i + 2, i + 3, i - 1, i - 2, x\}$ induces a banner. If x is not adjacent to some vertex in $\{i - 1, i, i + 1\}$, then there exists a vertex $j \in \{i - 1, i\}$ such that x distinguishes j and $j + 1$, and so $\{i - 3, i + 3, j, j + 1, x\}$ induces a banner. This proves that $S_i = \emptyset$ for $1 \leq i \leq 3$. \square

Claim 5. $S_0 = \emptyset$.

Proof (Proof of Claim 5). Suppose not. Let A be a connected component of S_0 . We first show that $N(A) \subseteq S_7$. By Claim 4 and the connectivity of G , some vertex $a \in A$ has a neighbor $n \in S_i$ for some $4 \leq i \leq 7$. If $n \in S_4 \cup S_5 \cup S_6$, then it can be readily seen that there exists an index i such that n is adjacent to $i - 3$ and $i + 3$ but not to some j where $j \in \{i - 1, i, i + 1\}$. Now $\{i - 3, i + 3, j, n, a\}$ induces a banner, a contradiction. This shows that $N(A) \subseteq S_7$. Since $N(A)$ is not a clique cutset by Lemma 1, it follows that A has at least two non-adjacent neighbors in S_7 . Let $x, y \in N(A)$ be non-adjacent. If there exists a vertex $a \in A$ such that a distinguishes x and y , then $\{1, 2, x, y, a\}$ induces a banner. Therefore, each vertex in A is either complete or anti-complete to $\{x, y\}$. Let A' be the set of vertices in A that are complete to $\{x, y\}$ and $A'' = A \setminus A'$. If A'' is not empty, then by the connectivity of A there exists an edge ab in A such that $a \in A'$ and $b \in A''$. Then $\{1, x, y, a, b\}$ induces a banner. Thus, $A'' = \emptyset$. This proves that $\{x, y\}$ is complete to A . Note that $\chi(A) \leq 4$ for otherwise $A \cup \{x, y\}$ has chromatic number at least 6. This contradicts that G is 6-vertex-critical. Since G is 6-vertex-critical, $G - V(A)$ admits a 5-coloring. Note that S_7 is monochromatic under this 5-coloring. Thus, one can extend this 5-coloring to G by assigning four colors that are not used on S_7 to A . This proves that $S_0 = \emptyset$. \square

Claim 6. For every $X \subseteq C$ with $4 \leq |X| \leq 6$, $S(X)$ is a clique.

Proof (Proof of Claim 6). Let $X \subseteq C$ be an arbitrary set with $4 \leq |X| \leq 6$. Since $4 \leq |X| \leq 6$, there exists an index $i \in C$ such that $i, i + 1 \in X$ but $i + 2 \notin X$. If $S(X)$ contains two non-adjacent vertices x and y , then $\{i, i + 1, i + 2, x, y\}$ induces a banner. Therefore, $S(X)$ is a clique. \square

Since G is 6-vertex-critical, it follows that $G = K_6$ if G contains a K_6 . Therefore, we may assume that G is K_6 -free. By Claim 6, it follows that $|S(X)| \leq 3$ for every $X \subseteq C$ with $4 \leq |X| \leq 6$. Therefore, $|S_4 \cup S_5 \cup S_6| \leq 3 \binom{7}{4} + \binom{7}{5} + \binom{7}{6} = 189$. By Claims 4 and 5, $G = C \cup S_4 \cup S_5 \cup S_6 \cup S_7$. Since G has no comparable vertices by Lemma 1, it follows from the pigeonhole principle that $|S_7| \leq 2^{|S_4 \cup S_5 \cup S_6|} \leq 2^{189}$. Therefore, the lemma holds. \square

4.3 Proof of Lemma 5

Proof. Let $C = 1, 2, 3, 4, 5$ be an induced C_5 . We partition $V \setminus C$ with respect to C . By (1) and (2), $S_1 = S_2 = \emptyset$ and $S(i - 2, i, i + 2) = \emptyset$ for $1 \leq i \leq 5$. By (3),

$S(i-1, i, i+1)$ and $S(C \setminus \{i\})$ are cliques. Since G is 6-vertex-critical, it follows that $G = K_6$ if G contains a K_6 . Therefore, we may assume that G is K_6 -free. This implies that each such clique has size at most three. Therefore, $|S_i| \leq 15$ for $i = 3, 4$. If S_5 is not bipartite, then S_5 contains a triangle or a five-cycle. This triangle or five-cycle together with C gives a 6-chromatic subgraph of G of order at most 10 and so $|V(G)| \leq 10$. Thus, we may assume that S_5 is bipartite.

Claim 7. $S_0 = \emptyset$.

Proof (Proof of Claim 7). Let A be a connected component of S_0 . By (4), A is anti-complete to $S_3 \cup S_4$. Therefore, $N(A) \subseteq S_5$. We first show that for any pair of non-adjacent vertices s and t in $N(A)$, $\{s, t\}$ is complete to A . Suppose that $a \in A$ is adjacent to s but not to t , then $\{1, 3, s, t, a\}$ induces a banner. Therefore, each vertex in A is either complete or anti-complete to $\{s, t\}$. If the set of vertices in A that is anti-complete to $\{s, t\}$ is not empty, then by the connectivity of A there is an edge $ab \in A$ such that a is anti-complete to $\{s, t\}$ and b is complete to $\{s, t\}$. Then $\{1, b, s, t, a\}$ induces a banner. Thus, $\{s, t\}$ is complete to A . Since $N(A)$ is not a clique by Lemma 1, $N(A)$ contains two non-adjacent vertices x and y . Since x and y are not comparable by Lemma 1, there exists a vertex d that distinguishes x and y , say d is adjacent to x but not to y . If $d \in N(A)$, then d is complete to A , since d and y are not adjacent. This implies that $\chi(A) \leq 3$. Since G is 6-vertex-critical, $G - V(A)$ admits a 5-coloring ϕ . Note that only two colors are used on S_5 under ϕ since $\chi(C_5) = 3$. Thus, we can extend this coloring to G by coloring A with three colors that are not used on S_5 , a contradiction. Therefore, $d \notin N(A)$. If $d \in S_5$, then $\{y, d, 1, 3, a\}$ induces a banner, where $a \in A$. If $d \notin S_5$, then there exists an index $i \in V(C)$ such that d is not adjacent to i . Then $\{x, y, i, a, d\}$ induces a banner. In either case we get a contradiction. \square

Claim 8. $|S_5| \leq 61 \cdot 2^{31}$.

Proof (Proof of Claim 8). Let $T_i = (X_i, Y_i)$ be the components of $G[S_5]$ for $1 \leq i \leq t$, where X_i and Y_i form the unique bipartition of T_i . By 3, each vertex in $S_3 \cup S_4$ is adjacent to all but at most one vertex in any stable set of S_5 . By the pigeonhole principle, if $t \geq 2|S_3 \cup S_4| + 2$ then there exist two components T_i and T_j such that $S_3 \cup S_4$ is complete to $T_i \cup T_j$. Without loss of generality, we may assume that $|V(T_i)| \leq |V(T_j)|$. Since G is 6-vertex-critical, $G - V(T_i)$ admits a 5-coloring ϕ . We then can extend ϕ to G by coloring T_i with at most two colors that are used on T_j under ϕ . This shows that $t \leq 2|S_3 \cup S_4| + 1 \leq 61$. It remains to bound each T_i . By Lemma 2, each T_i is $2P_2$ -free. Thus, we can order vertices in X_i as x_1, \dots, x_k such that $N_{T_i}(x_1) \subseteq \dots \subseteq N_{T_i}(x_k)$. By the pigeonhole principle, if $|X_i| > 2^{|S_3 \cup S_4|}$, then we can find two comparable vertices in X_i , which contradicts Lemma 1. Thus, $|X_i| \leq 2^{|S_3 \cup S_4|}$ and $|T_i| \leq 2^{|S_3 \cup S_4| + 1} \leq 2^{31}$. This proves the claim. \square

Recall that $|S_3| \leq 15$ and $|S_4| \leq 15$. Since $|V(G)| = |V(C)| + |S_0| + |S_3| + |S_4| + |S_5|$, the lemma follows immediately from Claims 7 and 8. \square

5 Conclusion

We have proved that there are finitely many 6-vertex-critical (P_5 , banner)-free graphs. Our result is one of the few results for finiteness of k -vertex-critical graphs when $k > 4$. Our result generalizes the previous result on the polynomial-time algorithm for 5-coloring P_5 -free graphs when restricted to banner-free graphs [15]. It is still open whether there are finitely many k -vertex-critical (P_5 , banner)-free graphs for fixed $k \geq 7$.

Acknowledgments. Qingqiong Cai was partially supported by National Natural Science Foundation of China (No. 11701297) and Open Project Foundation of Intelligent Information Processing Key Laboratory of Shanxi Province (No. CICIP2018005). Shenwei Huang is partially supported by the National Natural Science Foundation of China (11801284). Tao Li is partially supported by the National Natural Science Foundation (61872200) and the National Key Research and Development Program of China (2018YFB1003405, 2016YFC0400709). Yongtang Shi was partially supported by National Natural Science Foundation of China (Nos. 11771221, 11811540390), Natural Science Foundation of Tianjin (No. 17JCQNJC00300), and the China-Slovenia bilateral project “Some topics in modern graph theory” (No. 12-6).



References

1. Bondy, J.A., Murty, U.S.R.: Graph Theory. Springer, London (2008)
2. Bruce, D., Hoàng, C.T., Sawada, J.: A certifying algorithm for 3-colorability of P_5 -free graphs. In: Dong, Y., Du, D.-Z., Ibarra, O. (eds.) ISAAC 2009. LNCS, vol. 5878, pp. 594–604. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10631-6_61
3. Chudnovsky, M., Goedgebeur, J., Schaudt, O., Zhong, M.: Obstructions for three-coloring and list three-coloring H-free graphs. [arXiv:1703.05684](https://arxiv.org/abs/1703.05684) [math.CO]
4. Chudnovsky, M., Goedgebeur, J., Schaudt, O., Zhong, M.: Obstructions for three-coloring graphs with one forbidden induced subgraph. In: Proceedings of the Twenty-seventh Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1774–1783 (2016)
5. Chudnovsky, M., Robertson, N., Seymour, P., Thomas, R.: The strong perfect graph theorem. *Ann. Math.* **164**, 51–229 (2006)
6. Dhaliwal, H.S., Hamel, A.M., Hoàng, C.T., Maffray, F., McConnell, T.J.D., Panait, S.A.: On color-critical (P_5 , co- P_5)-free graphs. *Discrete Appl. Math.* **216**, 142–148 (2017)
7. Dirac, G.A.: Note on the colouring of graphs. *Mathematische Zeitschrift* **54**, 347–353 (1951)
8. Dirac, G.A.: A property of 4-chromatic graphs and some remarks on critical graphs. *J. Lond. Math. Soc.* **27**, 85–92 (1952)
9. Dirac, G.A.: Some theorems on abstract graphs. *Proc. Lond. Math. Soc.* **2**, 69–81 (1952)
10. Fouquet, J.L.: A decomposition for a class of (P_5 , $\overline{P_5}$)-free graphs. *Discrete Math.* **121**, 75–83 (1993)
11. Gallai, T.: Kritische Graphen I. *Publ. Math. Inst. Hungar. Acad. Sci.* **8**, 165–92 (1963)

12. Gallai, T.: Kritische Graphen II. Publ. Math. Inst. Hungar. Acad. Sci. **8**, 373–395 (1963)
13. Goedgebeur, J., Schaudt, O.: Exhaustive generation of k -critical \mathcal{H} -free graphs. J. Graph Theory **87**, 188–207 (2018)
14. Hell, P., Huang, S.: Complexity of coloring graphs without paths and cycles. Discrete Appl. Math. **216**, 211–232 (2017)
15. Hoàng, C.T., Kamiński, M., Lozin, V.V., Sawada, J., Shu, X.: Deciding k -colorability of P_5 -free graphs in polynomial time. Algorithmica **57**, 74–81 (2010)
16. Hoàng, C.T., Moore, B., Recoskiez, D., Sawada, J., Vatshelle, M.: Constructions of k -critical P_5 -free graphs. Discrete Appl. Math. **182**, 91–98 (2015)
17. Huang, S., Li, T., Shi, Y.: Critical (P_6 , banner)-free graphs. Discrete Appl. Math. (2018). <https://doi.org/10.1016/j.dam.2018.11.010>
18. Kaminski, M., Pstrucha, A.: Certifying coloring algorithms for graphs without long induced paths. [arXiv:1703.02485](https://arxiv.org/abs/1703.02485) [math.CO] (2017)
19. Kostochka, A.V., Yancey, M.: Ore’s conjecture on color-critical graphs is almost true. J. Combin. Theory Ser. B **109**, 73–101 (2014)
20. Ore, O.: The Four Color Problem. Academic Press, Cambridge (1967)
21. Randerath, B., Schiermeyer, I.: 3-colorability $\in \mathcal{P}$ for P_6 -free graphs. Discrete Appl. Math. **136**, 299–313 (2004)
22. West, D.: Introduction to Graph Theory. Prentice Hall, Upper Saddle River (2001)



An FPTAS for Stochastic Unbounded Min-Knapsack Problem

Zhihao Jiang^(✉)  and Haoyu Zhao^(✉) 

Institute for Interdisciplinary Information Sciences, Tsinghua University,
Beijing, China
{jzh16,zhaohy16}@mails.tsinghua.edu.cn

Abstract. In this paper, we study the stochastic unbounded min-knapsack problem (**Min-SUKP**). The ordinary unbounded min-knapsack problem states that: There are n types of items, and there is an infinite number of items of each type. The items of the same type have the same cost and weight. We want to choose a set of items such that the total weight is at least W and the total cost is minimized. The **Min-SUKP** generalizes the ordinary unbounded min-knapsack problem to the stochastic setting, where the weight of each item is a random variable following a known distribution and the items of the same type follow the same weight distribution. In **Min-SUKP**, different types of items may have different cost and weight distributions. In this paper, we provide an FPTAS for **Min-SUKP**, i.e., the approximate value our algorithm computes is at most $(1 + \epsilon)$ times the optimum, and our algorithm runs in $\text{poly}(1/\epsilon, n, \log W)$ time.

Keywords: Stochastic Knapsack · Renewal decision problem · Approximation algorithms

1 Introduction

In this paper, we study the stochastic unbounded min-knapsack problem (**Min-SUKP**). The problem is motivated by the following renewal decision problems introduced in [7]. A system (e.g., a motor vehicle) must operate for t units of time. A particular component (e.g., a battery) is essential for its operation and must be replaced each time it fails. There are n different types of replacement components, and every kind of items has infinite supplies. A type i replacement costs C_i and has a random lifetime with distribution depending on i . The problem is to assign the initial component and subsequent replacements from among the types to minimize the total expected cost of providing an operative component for the t units of time. Formally, we would like to solve the following **Min-SUKP** problem, defined as follows:

Problem 1 (stochastic unbounded min-knapsack). There are n types of items a_1, a_2, \dots, a_n . For an item of type a_i , the cost is a deterministic value c_i , and

the weight is random value X_i which follows a known distribution D_i with non-negative integer support. Let $D_i(j)$ denote $\Pr\{X_i \leq j\}$. Each type has infinite supplies, and the weight of each item is independent of the weight of the items of other types and other items of the same type. Besides, there is a knapsack with capacity W . Our objective is to insert items into the knapsack one by one until the total weight of items in the knapsack is at least W . The realized weight of an item is revealed to us as soon as it is inserted into the knapsack. What is the expected cost of the strategy that minimizes the expected total cost of the items we insert?

Remark 1. The above problem is the stochastic version of the ordinary unbounded min-knapsack problem. Comparing to the ordinary knapsack problem, there is an infinite number of items of each type, and the objective is to minimize the total cost (rather than maximize the total profit).

Remark 2. It can be shown that **Min-SUKP** is NP-hard. In [9], the authors mentioned that the unbounded knapsack problem (**UKP**) is NP-hard, and it can be easily shown that the unbounded min-knapsack is NP-hard, since there is a polynomial reduction between these 2 problems. The problem **Min-SUKP** is NP-hard since it is a generalization of unbounded min-knapsack.

Derman et al. [7] discussed **Min-SUKP** when the weight distributions of items are exponential and provided an exact algorithm to compute the optimal policy. Assaf [1] discussed **Min-SUKP** when the weight distributions of items have a common matrix phase type representation.

In this paper, we present a fully polynomial time approximation scheme (FPTAS) for this problem for general discrete distributions.

Roughly speaking, we borrow the idea of the FPTAS for the knapsack problem and the method for computing the distribution of the sum of random variables [16]. However, there are a few technical difficulties we need to handle. The outline of our algorithm is as follows. We first compute a constant factor approximation for the optimal cost (Sect. 2), and then we apply the discretization and a dynamic program based on the approximation value (Sect. 3). However, the dynamic program can only solve the problem in a restricted case where the cost for any item is ‘not too small’ (the cost of each item is larger than a specific value). To solve the whole problem, we consider a reduction from the general setting to the restricted setting and show that the error of the reduction is negligible (Sect. 4).

1.1 Related Work

The knapsack problem is a classical problem in combinatorial optimization. The classical knapsack problem (max-knapsack problem) is the following problem: Given a set of items with sizes and costs, and a knapsack with a capacity, our goal is to select some items and maximize the total cost of selected items with the constraint that the total size of selected items does not exceed the capacity of the knapsack.

The min-knapsack problem (**Min-KP**) [5] is a natural variant of the ordinary knapsack problem. In the min-knapsack problem, the goal is to minimize the total cost of the selected items such that the total size of the selected items is not less than the capacity of the knapsack. Although the min-knapsack problem is similar to the max-knapsack problem, a polynomial-time approximation scheme (PTAS) for the max-knapsack problem does not directly lead to a PTAS for the min-knapsack problem. For the (deterministic) min-knapsack problem, approximation algorithms with constant factors are given in [4, 5, 10]. Han and Makino [12] considered an online version of min-knapsack, that is, the items are given one-by-one over time.

There is also a line of work focusing on the FPTAS for unbounded knapsack problem (**UKP**). **UKP** is similar to the original 0-1 knapsack problem, except that there are infinite number of items of each type. The first FPTAS for **UKP** is introduced by [13], and they show an FPTAS by extending their FPTAS for 0-1 knapsack problem. Their algorithm runs in $O(n + \frac{1}{\epsilon^4} \log \frac{1}{\epsilon})$ time and needs $O(n + \frac{1}{\epsilon^3})$ space. Later, [15] showed an FPTAS with time complexity $O(n \log n + \frac{1}{\epsilon^2} (n + \log \frac{1}{\epsilon}))$ and space complexity $O(n + \frac{1}{\epsilon^2})$. In 2018, [14] presented an FPTAS that runs in $O(n \frac{1}{\epsilon^2} \log^3 \frac{1}{\epsilon})$ time and requires $O(n + \frac{1}{\epsilon} \log^2 \frac{1}{\epsilon})$ space.

However, in some applications, precisely knowing the size of each item is not realistic. In many real applications, we can only get the size distribution of a type of item. This problem leads to the stochastic knapsack problem (**SKP** [19]), which is a generalization of **KP**. In **SKP**, the cost of each item is deterministic, but the sizes of items are random variables with known distributions, and we get the realized size of an item as soon as it is inserted into the knapsack. The goal is to compute a solution policy which indicates the item we insert into the knapsack at a given remaining capacity. For the stochastic max-knapsack problem, an approximation with a constant factor was provided in the seminal work [6]. The current best approximation ratio for **SKP** is 2 [3, 18]. An $(1 + \epsilon)$ approximation with relaxed capacity (bi-criterion PTAS) is given in [2, 17]. Besides, Deshpande et al. [8] gave a constant-factor approximation algorithm for the stochastic min-knapsack.

Gupta et al. [11] considered a generalization of **SKP**, where the cost of items may be correlated, and we can cancel an item during its execution in the policy. Cancelling an item means we can set a bounding size each time we want to insert an item, we cancel the item if the realized size of the item is larger than the bounding size. When we cancel an item, the size of the item is equal to the bounding size, and the cost of the item is zero. This generalization is referred to as Stochastic Knapsack with Correlated Rewards and Cancellations (**SK-CC**). Gupta et al. [11] gave a constant-factor approximation for **SK-CC** based on LP relaxation. A bicriterion PTAS for **SK-CC** is provided in [17].

1.2 Preliminary

Proposition 1. *Without the loss of generality, we can assume that the support of D_i , which is the weight distribution of an item of type i , has positive integer support.*

We skip the proof of Proposition 1. Please see the proof in Appendix B.

From now on, we can suppose that each type of item has weight distribution with positive integer support.

In **Min-SUKP**, the optimal item added can be determined by the remaining capacity. Let OPT_w denote the expected cost of the optimal strategy when the remaining size is w . We can assume that the support of D_i is $\{0, 1, \dots, W\}$. Let $OPT_0 = OPT_{-1} = \dots = OPT_{-W+1} = 0$. Define $d_i(j) = D_i(j) - D_i(j-1) = \Pr\{X_i = j\}$. From the dynamic program, we have pseudo-polynomial time Algorithm 1 that can compute the exact optimal value.

Algorithm 1. Pseudo-polynomial Time Algorithm

```

1:  $OPT_i \leftarrow 0$  for  $-W + 1 \leq i \leq 0$ 
2: for  $i = 1 \rightarrow W$  do
3:    $OPT_i = \min_{j=1}^n \left( c_j + \sum_{k=1}^W d_j(k) \cdot OPT_{i-k} \right)$ 
return  $OPT_W$ 

```

Algorithm 1 runs in **poly** (n, W) time.

In this paper, we show an FPTAS to compute OPT_W . Our algorithm runs in **poly** $(\frac{1}{\epsilon}, n, \log W)$ time and return OPT'_W , which is an approximation for OPT_W , such that $(1 - \epsilon)OPT_W \leq OPT'_W \leq (1 + \epsilon)OPT_W$. We assume that there is an oracle \mathcal{A} such that we can call \mathcal{A} to get $D_i(j) = \Pr\{X_i \leq j\}$. Since we require that our algorithm runs in **poly** $(\frac{1}{\epsilon}, n, \log W)$ time, our algorithm can call the oracle for at most **poly** $(\frac{1}{\epsilon}, n, \log W)$ times.

2 A Constant Factor Estimation

In this section, we show that there is a constant factor approximation for the optimal value. This constant factor approximation serves to estimate the optimal value roughly, and our FPTAS uses the standard discretization technique based on this rough estimation.

Define $b_i = \frac{c_i}{E[X_i]}$. When we insert an item of type i , the expected weight is $E[X_i]$, and the cost is $c_i = b_i E[X_i]$. Suppose $m = \arg \min_i b_i$, and we will show that $2b_m W$ is a constant approximation for the optimal value OPT_W . Formally, we have the following lemma,

Lemma 1. *For all $-W + 1 \leq w \leq W$, $b_m w \leq OPT_w \leq b_m(w + W)$, where $m = \arg \min_i b_i$.*

This lemma can be proved by induction, and please see Appendix C for its formal proof.

Specifically, when $w = W$, we get $b_m W \leq OPT_W \leq 2b_m W$ directly from the above lemma. However, when computing b_m , we need to enumerate the support.

To avoid expensive enumeration, we can compute $E[X_i]$ approximatively. We round the realized weight x_i into $2^{\lceil \log_2 x_i \rceil}$. Just let

$$\bar{E}[X_i] = \sum_{j=1}^W d_i(j) 2^{\lceil \log_2 j \rceil} = D_i(1) + \sum_{j=0}^{\lfloor \log W \rfloor} (2^j \cdot (D_i(2^{j+1}) - D_i(2^j))).$$

We have $\frac{E[X_i]}{2} \leq \bar{E}[X_i] \leq E[X_i]$, since $\frac{x_i}{2} \leq 2^{\lceil \log_2 x_i \rceil} \leq x_i$.

Let $\overline{OPT}_W = 2W \cdot \min_i \frac{c_i}{\bar{E}[X_i]}$. From the previous argument, we have $2b_m W \leq \overline{OPT}_W \leq 4b_m W$, which means $OPT_W \leq \overline{OPT}_W \leq 4OPT_W$.

Let $T = \frac{1}{4} \overline{OPT}_W$. We have $\frac{1}{4} OPT_W \leq T \leq OPT_W$. T is the estimation of OPT_W .

3 FPTAS Under Certain Assumption

In this section, we discuss **Min-SUKP** under the following assumption.

Definition 1 (Cheap/Expensive type). Let $\theta = \frac{\epsilon}{10n}$. We call type i is an expensive type if $c_i \geq \theta T$, otherwise we call type i is a cheap type.

Assumption 1. We assume all the types are expensive.

And we give an algorithm with approximation error at most ϵT in this section under Assumption 1.

In general, our algorithm for **Min-SUKP** is inspired from the FPTAS of the ordinary knapsack problem [20]. We define $\hat{f}_c = \max\{w | OPT_w \leq c\}$, and compute the approximation for \hat{f} . However, the support of \hat{f} is the set of real numbers. So we discretize \hat{f} and only compute the approximation for $\hat{f}_{i\delta T}$ for all $i \leq \lceil \frac{1}{\delta} \rceil + 1$, where i is non-negative integer and $\delta = \frac{\epsilon^2}{100n}$. In our algorithm, we use dynamic programming to compute f_i , which is the approximation for $\hat{f}_{i\delta T}$. Then we use f_i to get an approximate value of OPT_W . Since $\hat{f}_{i\delta T}$ is monotonically increasing with respect to i , we can find the smallest i such that $f_i \geq W$ and return the value $i\delta T$ as the approximate value of OPT_W .

Now we show how to compute f_i . First, suppose that $\hat{f}_{i\delta T} = w^*$, and from the dynamic programming, we have

$$OPT_{w^*} = \min_k \left\{ c_k + \sum_{j=1}^W d_k(w^* - j) OPT_j \right\}.$$

Since OPT_w is non-decreasing while w is increasing, recall $\hat{f}_c = \max\{w | OPT_w \leq c\}$, and we get,

$$\begin{aligned} w^* &= \max \left\{ w' \mid \exists k, c_k + \sum_{j=1}^W d_k(w' - j) OPT_j \leq i\delta T \right\} \\ &= \max \left\{ w' \mid \exists k, c_k + \sum_{j=1}^{w'-1} d_k(w' - j) OPT_j \leq i\delta T \right\}. \end{aligned}$$

Algorithm 2. The Dynamic Program for Computing approximate answer for **Min-SUKP** under Assumption 1

```

1: Let  $\delta = \frac{\epsilon^2}{100n}$ .
2:  $f_0 \leftarrow 0$ 
3: for  $i = 1 \rightarrow \lceil \frac{1}{\delta} \rceil + 1$  do
4:   Compute  $f_i$  using binary search according to Algorithm 3
5:   if  $f_i \geq W$  then
6:     return  $\hat{V} := i\delta T$ 

```

Algorithm 3. Given w , judge whether $f_i \geq w$ (whether $g_w \leq i\delta T$)

```

1: for  $j = 1 \rightarrow n$  do
2:   for  $m = 0 \rightarrow i - 2$  do
3:      $P_m = Pr[X_j \in [w - f_{m+1}, w - f_m]]$  ▷ by Binary search from oracle
4:      $P_{i-1} = Pr[X_j \in [1, w - f_{i-1}]]$ 
5:      $g_w \leftarrow c_j + \sum_{m=0}^{i-1} P_m(m+1)\delta T$  ▷ Equation (1)
6:     if  $g_w \leq i\delta T$  then return true
   return false

```

Define $\hat{g}_w := j\delta T$ for all $\hat{f}_{j-1} < w \leq \hat{f}_j$. Then \hat{g}_w is the rounding up discretization value of OPT_w , and we can approximately compute w^* (let \hat{w} denote the approximate value) by

$$\hat{w} = \max \left\{ w' \mid \exists k, \left(c_k + \sum_{j=1}^{w'-1} d_k(w' - j)\hat{g}_j \right) \leq i\delta T \right\}.$$

However, we do not have \hat{g} during the computation. Instead, we use the following quantity to approximate \hat{g} . Given f_0, \dots, f_{i-1} , define $g_w := j\delta T$ for all $f_{j-1} < w \leq f_j$ where $j \leq i - 1$, and define $g_w = i\delta T$ for all $w > f_{i-1}$. Then we have

$$f_i = \max \left\{ w' \mid \exists k, \left(c_k + \sum_{j=1}^{w'-1} d_k(w' - j)g_j \right) \leq i\delta T \right\}. \quad (1)$$

Remark 3. When we compute f_i , we have already gotten f_0, f_1, \dots, f_{i-1} .

To compute f_i , we use binary search to guess $f_i = w'$ and accept the largest w' that satisfies the constraint in (1).

The pseudo-code of our algorithm is shown in Algorithm 2. The detailed version of the pseudo-codes is presented in Appendix A.

In details, we enumerate i and compute f_i until f_i reaches the weight lower limit W . To compute f_i , we use binary search starting with $L = 0, R = W$. In each step of binary search, let $w = (L + R)/2$ and compute g_w , and decide to recur in which half according to the relation between g_w and $i\delta T$, until $L = R$ which means $f_i = L = R$.

To quantify the approximation error by Algorithm 2, we have the following theorem.

Theorem 1. *The output \hat{V} of Algorithm 2 satisfies $(1-\delta)(1-\frac{\epsilon}{10})\hat{V} \leq OPT_W \leq \hat{V}$.*

Generally speaking, this results can be shown in 2 steps: First, we will show that the real optimal value is upper bounded by the value computed in our algorithm, and next, we will show that under Assumption 1, the difference between the value computed in Algorithm 2 and the real optimal value is upper bounded by a small value. Given these two results, we can prove Theorem 1. Please see Appendix D for the formal proof of Theorem 1.

From the above theorem, we know that the output \hat{V} of Algorithm 2 is a $(1 + \epsilon)$ -approximation for OPT_W .

4 FPTAS in the General Case

In the previous section, we show that there is an FPTAS of **Min-SUKP** under Assumption 1 (when all the types are expensive). In this section, we remove Assumption 1 and show that there is an FPTAS of **Min-SUKP**. We will first present the general idea of our algorithm.

Our Ideas: If we use the algorithm in the last section to compute in general case, the error will not be bounded. The key reason is that we may insert lots of items of cheap types. One idea is, we can bundle lots of items in the same cheap type p into bigger items (an induced type p'), such that p' is expensive. Then we replace type p by the new type p' . Now, we can use the algorithm in the last section. However, we can only use bundled items even if we only want to use one item of a certain cheap item. Luckily, using some extra items of cheap items does not weaken the policy very much.

The remaining problem is, how to compute the distribution of many items of type p ? For example, we always use $e_p = 2^k$ items of type p each time. We discretize the weight distribution X_p , and use doubling trick to compute the approximate distributions for $X_{p,1}, \sum_{i=1}^2 X_{p,i}, \sum_{i=1}^4 X_{p,i}, \dots$ one by one, where $X_{p,i}$ are independent to each other and follow the same distribution of X_p . We can show that, using the approximation distributions in the computation will not lead to much error.

4.1 Adding Limitations to Strategy

For type p , if $c_p < \theta T$ ($\theta = \frac{\epsilon}{10n}$ as defined in the previous section), then there exists $e_p = 2^{k_p}, k_p \in Z$ such that $e_p c_p \in [\theta T, 2\theta T]$. For convenience, if $c_p \geq \theta T$, we denote $e_p = 1$. We have the following restriction to the strategy.

Definition 2 (Restricted strategy). *A strategy is called restricted strategy, if for all type p , the total number of items of type p we insert is always a multiple of e_p .*

If we know that for all type p , the total number of items of type p is always a multiple of e_p , we hope that each time we use an item of type p , we will use e_p of them together. This leads to the following definition.

Definition 3 (Block strategy). *A strategy is called block strategy, if we always insert a multiple of e_p number of items of type p together.*

The following theorem shows that, adding limitation to the strategy will not affect the optimal value too much.

Theorem 2. *Suppose the expected cost of the best block strategy is $OPT_W^{(b)}$, then $OPT_W \leq OPT_W^{(b)} \leq OPT_W + \frac{\epsilon T}{5}$.*

Because of the space limitation, we will present the proof sketch below. For the formal proof of Theorem 2, please see Appendix E.

Proof (Proof sketch). The proof Theorem 2 is divided into 2 parts. The first part shows that the optimal value for the original problem does not differ much from the optimal value with restricted strategy (see Definition 2), and the second part shows that the optimal value with restricted strategy is the same as the optimal value with block strategy (see Definition 3). The first part is simple since we can add some item after following the optimal strategy in the original problem. The second part follows from the intuition that if we must use an item in the future, it is good to use it right now.

4.2 Computing the Summation Distribution of Many Items of the Same Type

In the last part, we define block strategy by adding a constraint to the ordinary strategy. And we find the expected cost of the optimal block strategy is close to that of the optimal strategy.

The block strategies conform to Assumption 1 in Sect. 3. If we know the distribution of the total weight of e_p items of type p , we can compute the approximate optimal expected cost by Algorithm 2. In this part, we give an algorithm which approximately computes the distribution of the total weight of e_p items of type p .

Due to the space limitation, we present our algorithm in this section, and we put the analysis of our algorithm into the appendix (see Appendix F). To present our idea, we need the following definitions.

Definition 4 (Distribution Array). *For a random variable X with positive integer support, we use $X[i]$ to denote the probability that $X \geq i$, i.e. $X[i] = \Pr\{X \geq i\}$, and we use an array $\text{Dist}(X) := (X[1], X[2], \dots, X[W])$ to denote the distribution. We call $\text{Dist}(X)$ the distribution array of variable X .*

Remark 4. From the definition, we know that $\text{Dist}(X)$ is a non-increasing array. Besides, in the definition, $\text{Dist}(X)$ has only W elements since we only care $X[i]$ when $i \leq W$.

Definition 5. For any non-increasing array $D = (D_1, D_2, \dots, D_W)$ of length W , if $D_1 \leq 1$ and $D_W \geq 0$, there is a random variable X such that $\text{Dist}(X) = D$. We say that X is the variable corresponding to distribution array D , denoted by $\text{Var}(D) := X$.

Suppose $\{Y_i\}_{i \geq 1}$ are identical independent random variables with distribution array $\text{Dist}(X_p)$. Let S_i denote $\sum_{j=1}^i Y_j$ and $\text{Dist}(S_i)$ denote the corresponding distribution array. We want to compute the distribution array of S_{e_p} and we have the following equations,

$$\Pr\{S_{2i} = w\} = \sum_{j=1}^{w-1} (\Pr\{S_i = j\} \cdot \Pr\{S_i = w - j\}), \forall 1 \leq w \leq W, \quad (2)$$

$$S_{2i}[w] = \Pr\{S_i \geq w\} + \sum_{j=1}^{w-1} (\Pr\{S_i = j\} \cdot \Pr\{S_i \geq w - j\}) \quad (3)$$

$$= S_i[w] + \sum_{j=1}^{w-1} ((S_i[j] - S_i[j + 1]) \cdot S_i[w - j]). \quad (4)$$

Note that S_{2i} can be computed from S_i , so we only need to compute $S_1, S_2, S_4, \dots, S_{e_p}$ successively (recall that $e_p = 2^{k_p}$ where $k_p \in \mathbb{Z}$). Note that S_1 could be got from the oracle.

However, computing the exact distribution of S_{2^i} is slow (needs at least $\text{poly}(W)$ time), so we compute an approximate value of S_i . To introduce our method which approximately computes the distribution, we need the following definitions.

Definition 6 (η -Approximate Array). Given a positive real number η , for distribution array $A = (a_1, a_2, \dots, a_m)$, define $A' = (a'_1, a'_2, \dots, a'_m)$ as the η -approximate array of A , where for all $i \in [m]$,

$$a'_i = (1 + \eta)^{\lceil \log_{1+\eta} a_i \rceil}, a_i > (1 + \eta)^{-\zeta}.$$

Definition 7 ((ζ, η) -Approximate Array). Given positive real numbers ζ, η , for distribution array $A = (a_1, a_2, \dots, a_m)$, define $A' = (a'_1, a'_2, \dots, a'_m)$ as the (ζ, η) -approximate array of A , where for all $i \in [m]$,

$$a'_i = \begin{cases} (1 + \eta)^{\lceil \log_{1+\eta} a_i \rceil}, & a_i > (1 + \eta)^{-\zeta} \\ (1 + \eta)^{-\zeta}, & a_i \leq (1 + \eta)^{-\zeta} \end{cases}.$$

Definition 8 ((ζ, η) -Approximation). For random variable X , suppose distribution array D is (ζ, η) -approximate array of $\text{Dist}(X)$. Define $\text{Var}(D)$ as the (ζ, η) -approximation of X .

Remark 5. The (ζ, η) -Approximation of a random variable is still a random variable. And for any random variable X with integer support in $[1, W]$, the (ζ, η) -approximation of X has at most $\lceil \zeta \rceil$ different possible values.

Let $\eta = \frac{\epsilon}{10 \log W}$ and $\zeta = \log_{1+\eta} \frac{W}{\eta}$, and our algorithm is shown as following: We first compute (ζ, η) -approximation of S_1 which is denoted by B_1 . Then for all $i \in \lceil \log_{e_p} \rceil$, we compute the distribution array of B'_{2^i} , which is the summation of independent $B_{2^{i-1}}$ and $B_{2^{i-1}}$. Then we compute B_{2^i} which is the (ζ, η) -approximation for B'_{2^i} . Finally, we can get B_{e_p} which is an approximate random variable of S_{e_p} .

When we compute the summation of $B_{2^{i-1}}$ and $B_{2^{i-1}}$, as there are at most $O(\zeta)$ different values in $\text{Dist}(B_{2^{i-1}})$, there are at most $O(\zeta)$ values w such that $\Pr\{B_{2^{i-1}} = w\} > 0$. Based on the previous argument, we can enumerate w_1 and w_2 such that $\Pr\{B_{2^{i-1}} = w_1\} > 0$ and $\Pr\{B_{2^{i-1}} = w_2\} > 0$. In the end, we sort each $\Pr\{B_{2^{i-1}} = w_1\} \cdot \Pr\{B_{2^{i-1}} = w_2\}$ by the value $w_1 + w_2$ and arrange them to get the distribution array $\text{Dist}(B'_{2^i})$. This shows that we can compute the approximate distribution in $O(\zeta^2 \log \zeta)$ time.

Formally, we have Algorithm 4 to compute B_{e_p} .

Algorithm 4. Computing Approximate Distribution of S_{e_p}

- 1: Let $\eta = \frac{\epsilon}{10 \log W}$ and $\zeta = \log_{1+\eta} \frac{W}{\eta}$.
 - 2: Let B_1 be the (ζ, η) -approximation for S_1 . Compute $\text{Dist}(B_1)$ according to the oracle
 - 3: **for** $i = 1 \rightarrow \log_2 e_p$ **do**
 - 4: Let B'_{2^i} be the summation of $B_{2^{i-1}}$ and $B_{2^{i-1}}$. Compute $\text{Dist}(B'_{2^i})$.
 - 5: Compute $\text{Dist}(B_{2^i})$, which is the (ζ, η) -approximation for B'_{2^i} .
 - 6: **return** $\text{Dist}(B_{e_p})$
-

Before we state the main theorem that bounds the approximation error of our algorithm, we combine the full procedure and get our final Algorithm 5 for **Min-SUKP**.

Algorithm 5. Algorithm for **Min-SUKP**

- 1: For each type p , let $S_{e_p}^p$ be the summation of e_p i.i.d. variables with distribution X_p . Compute approximate distribution Y_p of $S_{e_p}^p$ by Algorithm 4.
 - 2: For each item type p , construct new type p' with expected cost $c_p e_p$ and weight distribution Y_p .
 - 3: Let W and all the new types be the input, and get return value \hat{V} of Algorithm 2.
 - 4: **return** \hat{V} .
-

Then, we have our main theorem, which discusses the approximation error of Algorithm 5.

Theorem 3. *The output \hat{V} of Algorithm 5 satisfies*

$$(1 - \epsilon)OPT_W \leq \hat{V} \leq (1 + \epsilon)OPT_W.$$

To prove this theorem, we first show $\text{Dist}(B_{e_p})$ in Algorithm 4 is approximation of $\text{Dist}(A_{e_p})$, by constructing another strategy C_{e_p} which is strictly better than B_{e_p} and the expected cost of C_{e_p} is closed to the expected cost of A_{e_p} (induction is used). Then we combine all the errors in Algorithm 5 and prove that Algorithm 5 is FPTAS of **Min-SUKP**. For details, please see Appendix F.

4.3 Time Complexity

Our algorithm runs in $\text{poly}(n, \log W, \frac{1}{\epsilon})$. Combined with Theorem 3, Algorithm 5 is an FPTAS for **Min-SUKP**. The theorem for the time complexity of Algorithm 5 is stated as follow,

Theorem 4. *Algorithm 5 runs in polynomial time and thus is an FPTAS for **Min-SUKP**. More specifically, Algorithm 5 has time complexity*

$$O\left(\frac{n \log^6 W}{\epsilon^3} + \frac{n^3 \log W}{\epsilon^4}\right).$$

This theorem can be proved by recalling the parameters we have set, counting for the number of each operation, and expanding the parameters as n , $\log W$ and $\frac{1}{\epsilon}$. Please see Appendix G for the formal proof.

5 Conclusions and Further Work

We obtain the first FPTAS for **Min-SUKP** in this paper. We focus on computing approximately the optimal value, but our algorithms and proofs immediately imply how to construct an approximate strategy in polynomial time.

There are some other directions related to **Min-SUKP** which are still open. It would be interesting to design a PTAS (or FPTAS) for the 0/1 stochastic minimization knapsack problem, the 0/1 stochastic (maximization) knapsack problem and the stochastic unbounded (maximization) knapsack problem. Hopefully, our techniques can be helpful in solving these problem.

Acknowledgement. The authors would like to thank Jian Li for several useful discussions and the help with polishing the paper. The research is supported in part by the National Basic Research Program of China Grant 2015CB358700, the National Natural Science Foundation of China Grant 61822203, 61772297, 61632016, 61761146003, and a grant from Microsoft Research Asia.

References

1. Assaf, D.: Renewal decisions when category life distributions are of phase-type. *Math. Oper. Res.* **7**(4), 557–567 (1982)
2. Bhalgat, A., Goel, A., Khanna, S.: Improved approximation results for stochastic knapsack problems. In: *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, Philadelphia (2011)

3. Bhargat, A.: A $(2 + \epsilon)$ -approximation algorithm for the stochastic knapsack problem. Manuscript (2012)
4. Carnes, T., Shmoys, D.: Primal-dual schema for capacitated covering problems. In: Lodi, A., Panconesi, A., Rinaldi, G. (eds.) IPCO 2008. LNCS, vol. 5035, pp. 288–302. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68891-4_20
5. Csirik, J., Frenk, J.B.G., Labbe, M., Zhang, S.: Heuristics for the 0-1 min-knapsack problem. *Acta Cybern.* **10**(1–2), 15–20 (1991)
6. Dean, B.C., Goemans, M.X., Vondrak, J.: Approximating the stochastic knapsack problem: the benefit of adaptivity. In: Annual IEEE Symposium on Foundations of Computer Science, pp. 208–217. IEEE Computer Society, Los Alamitos (2004)
7. Derman, C., Lieberman, G.J., Ross, S.M.: A renewal decision problem. *Manag. Sci.* **24**(5), 554–561 (1978)
8. Deshpande, A., Hellerstein, L., Kletenik, D.: Approximation algorithms for stochastic submodular set cover with applications to boolean function evaluation and min-knapsack. *ACM Trans. Algorithms* **12**(3), 28 (2016)
9. Garey, M.R., Johnson, D.S.: *Computers and Intractability*, vol. 29. W. H. Freeman, New York (2002)
10. Guntzer, M.M., Jungnickel, D.: Approximate minimization algorithms for the 0/1 knapsack and subset-sum problem. *Oper. Res. Lett.* **26**(2), 55–66 (2000)
11. Gupta, A., Krishnaswamy, R., Molinaro, M., Ravi, R.: Approximation algorithms for correlated knapsacks and non-martingale bandits. In: IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, 22–25 October 2011, pp. 827–836 (2011)
12. Han, X., Makino, K.: Online minimization knapsack problem. In: Bampis, E., Jansen, K. (eds.) WAOA 2009. LNCS, vol. 5893, pp. 182–193. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12450-1_17
13. Ibarra, O.H., Kim, C.E.: Fast approximation algorithms for the knapsack and sum of subset problems. *J. ACM (JACM)* **22**(4), 463–468 (1975)
14. Jansen, K., Kraft, S.E.: A faster FPTAS for the unbounded knapsack problem. *Eur. J. Comb.* **68**, 148–174 (2018)
15. Kellerer, H., Pferschy, U., Pisinger, D.: Multidimensional knapsack problems. In: Kellerer, H., Pferschy, U., Pisinger, D. (eds.) *Knapsack Problems*, pp. 235–283. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24777-7_9
16. Li, J., Shi, T.L.: A fully polynomial-time approximation scheme for approximating a sum of random variables. *Oper. Res. Lett.* **42**(3), 197–202 (2014)
17. Li, J., Yuan, W.: Stochastic combinatorial optimization via poisson approximation. In: Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing, STOC 2013, pp. 971–980. ACM, New York (2013)
18. Ma, W.: Improvements and generalizations of stochastic knapsack and multi-armed bandit approximation algorithms. In: Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (2014)
19. Ross, K.W., Tsang, D.H.: The stochastic knapsack problem. *IEEE Trans. Commun.* **37**(7), 740–747 (1989)
20. Sahni, S.: Approximate algorithms for 0/1 knapsack problem. *J. ACM* **22**(1), 115–124 (1975)



The Inapproximability of k -DominatingSet for Parameterized AC^0 Circuits

Wenxing Lai^(✉)

Shanghai Key Laboratory of Intelligent Information Processing,
School of Computer Science, Fudan University, Shanghai, China
wenxing.lai@outlook.com

Abstract. In [4], Chen and Flum showed that any FPT-approximation of the k -CLIQUE problem is not in para-AC^0 and the k -DOMINATINGSET (k -DOMSET) problem could not be computed by para-AC^0 circuits. It is natural to ask whether $f(k)$ -FPT-approximation of the k -DOMSET problem is in para-AC^0 for some computable function f .

Very recently [13, 20] showed that assuming $\text{W}[1] \neq \text{FPT}$, the k -DOMSET cannot be approximated by FPT algorithms. We observe that the constructions in [13] can be carried out in para-AC^0 , and thus we prove that para-AC^0 circuits could not approximate this problem with ratio $f(k)$ for any computable function f . Moreover, under the hypothesis that the 3-CNF-SAT problem cannot be computed by constant-depth circuits of size $2^{\varepsilon n}$ for some $\varepsilon > 0$, we show that constant-depth circuits of size $n^{o(k)}$ cannot distinguish graphs whose dominating numbers are either $\leq k$ or $> \sqrt{\frac{k \log n}{3 \log \log n}}$. However, we find that the hypothesis may be hard to settle by showing that it implies $\text{NP} \not\subseteq \text{NC}^1$.

Keywords: Parameterized AC^0 · Dominating set · Inapproximability

1 Introduction

The dominating set problem, or equivalently the set cover problem, firstly shown to be NP-complete in [12], is often regarded as one of the most classical and important problems in computational complexity. Unless $\text{P} = \text{NP}$, we do not expect to solve this problem and its optimization variant in polynomial time. One way to handle NP-hard problems is to use approximation algorithms. One key measurement of an approximation algorithm for dominating set problem is by its approximation ratio, i.e., the ratio between the size of the solution output by the algorithm and the size of the minimum dominating set. It is known that greedy algorithms can achieve approximation ratio $\approx \ln n$ [5, 11, 14, 22, 23]. After a long line of works [2, 8, 15, 16, 18], this is matched by the lower bound by Dinur and Steurer [6], who followed the construction in [8], showing that for every $\varepsilon > 0$, we could obtain a $(1 - \varepsilon) \ln n$ -approximation for this problem unless $\text{P} = \text{NP}$.

Besides approximation, another widely-considered technique to circumvent the intractability of NP-hardness is parameterization. If we take the minimum solution size k as a parameter, then the brute-force algorithm can solve this problem in n^{k+1} time. However, it is proved recently that, assuming $\text{FPT} \neq \text{W}[1]$, for any computable function f , there is no $f(k)$ -FPT-approximation algorithm, that is, there is no approximation algorithm running in FPT-time and with ratio $f(k)$ [3, 13, 20].

Circuit complexity was thought to be a promising direction to solve P vs. NP. Though it has been long known that some problems like PARITY are not in AC^0 [1, 9, 10], proving non-uniform lower bounds for functions in nondeterministic complexity classes such as NP, $\text{NQP} = \text{NTIME}[n^{\log^O(1)n}]$ or NEXP is a well-known challenge. After Williams proving that NEXP does not have $n^{\log^O(1)n}$ -size $\text{ACC} \circ \text{THR}$ (ACC composed with a layer of linear threshold gates at the bottom) in [24, 25], Murray and Williams showed that for every k, d and m there is an e and a problem in $\text{NTIME}[n^{\log^e n}]$ which does not have depth- d $n^{\log^k n}$ -size $\text{AC}[m]$ circuits with linear threshold gates at the bottom layer [17].

In [19], Rossman showed that the k -CLIQUE problem has no bounded-depth and unbounded fan-in circuits of size $O(n^{k/4})$, which may be viewed as an AC^0 version of $\text{FPT} \neq \text{W}[1]$. Chen and Flum [4] showed that any FPT-approximation of the k -CLIQUE problem is not in para-AC^0 . The parameterized circuit complexity class para-AC^0 introduced in [7] as the AC^0 analog of the class FPT, is the class of parameterized problems computed by constant-depth circuits of size $f(k)\text{poly}(n)$ for some computable function f . In the same paper, based on Rossman's result, they also showed that the k -DOMSET problem could not be computed in para-AC^0 . This brings us to the main question addressed in our work: *Is there a computable function f such that the $f(k)$ -approximation of k -DOMSET is in para-AC^0 ?* Furthermore, since we could brute-forcefully enumerate every k tuple of vertices by depth-3 circuits of size $O(n^{k+1})$, we might wonder whether it is possible to have a computable function f such that the $f(k)$ -approximation of k -DOMSET could be computed by constant-depth circuits of size $n^{o(k)}$.

Our Work. In this paper, we show that any FPT-approximation of the k -DOMSET problem is not in para-AC^0 . Furthermore, under the hypothesis that constant-depth circuits of size $2^{o(n)}$ could not compute 3-CNF-SAT, the constant-depth version of the Exponential Time Hypothesis (ETH), there is no computable function f such that the $f(k)$ -approximation of k -DOMSET could be computed by constant-depth circuits of size $n^{o(k)}$. Theorems 1 and 2 are direct consequences of Theorems 3 and 4, respectively.

Theorem 1. *Given a graph G with n vertices, there is no constant-depth circuits of size $f(k)n^{o(\sqrt{k})}$ for any computable function f which distinguish between*

- the size of the minimum dominating set is at most k ,
- the size of the minimum dominating set is greater than $(\frac{\log n}{\log \log n})^{1/\binom{k}{2}}$.

Note that this theorem implies the nonexistence of para-AC^0 circuits which $f(k)$ -approximates the k -DOMSET problem for any computable function f . This

is because if there is an $f(k)$ -approximation para- AC^0 circuit $C_{n,k}$ whose size is $g(k)\text{poly}(n)$, we could construct a constant-depth para- AC^0 circuit $C'_{n,k}$ to distinguish the size of the minimum dominating set is at most k or greater than $(\frac{\log n}{\log \log n})^{1/\binom{k}{2}}$ as follows. Compare $f(k)$ and $(\frac{\log n}{\log \log n})^{1/\binom{k}{2}}$ —if $f(k)$ is smaller, we let $C'_{n,k}$ be $C_{n,k}$; otherwise, since $f(k) \geq (\frac{\log n}{\log \log n})^{1/\binom{k}{2}}$, we let $C'_{n,k}$ be the circuit which brute-force computes the size of a minimum dominating set with the depth-3 circuit of size $n^{k+1} \leq 2^{kf(k)^{k^3}}$, which is still a para- AC^0 circuit.

Hypothesis 1. *There exists $\delta > 0$ such that no constant-depth circuits of size $2^{\delta n}$ can decide whether the 3-CNF-SAT instance φ is satisfiable, where n is the number of variables of φ .*

Theorem 2. *Assuming Hypothesis 1, given a graph G with n vertices, there is no constant-depth circuits of size $f(k)n^{o(k)}$ for any computable function f which distinguish between*

- the size of the minimum dominating set is at most k ,
- the size of the minimum dominating set is greater than $\sqrt[k]{\frac{\log n}{3 \log \log n}}$.

Though Hypothesis 1 seems much weaker than ETH (ETH implies the nonexistence of uniform circuits of size $2^{\delta n}$ and any depth which could compute 3-CNF-SAT problem), we show that the hypothesis is hard to settle by proving that it implies $\text{NP} \not\subseteq \text{NC}^1$, which, believed to be true, remains open for decades and whether whose weaker version, $\text{NP} \not\subseteq \text{ACC} \circ \text{THR}$, holds or not is still unknown.

2 Preliminaries

We denote by \mathbb{N} the set of nonnegative integers. For each $n \in \mathbb{N}$, we define $[n] := \{1, \dots, n\}$. For any set A and $k \in \mathbb{N}$, we let $\binom{A}{k} := \{B \subseteq A \mid |B| = k\}$ be the set of subsets with exactly k elements of A . For a sequence of bits b , we let $b[l]$ be the l -th bit of b .

For a graph $G = (V, E)$, the set of vertices of G is denoted by V_G and the set of edges is denoted by E_G ; for a vertex $v \in V_G$, we let $N_G(v) := \{u \in V_G \mid \{u, v\} \in E_G\}$ be the neighbors of v . Since a graph G is represented using a binary string, we express the bit of the edge $\{u, v\}$ by $\text{bit}_G\{u, v\}$.

Problem Definitions. The decision problems studied in this paper are listed below.

- In the k -DOMINATINGSET (k -DOMSET) problem, our goal is to decide if there is a dominating set of size k in the given graph G .
- In the k -SETCOVER problem, we are given a bipartite graph $I = (S, U, E)$ and the goal is to decide whether there is a subset X of S with cardinality k such that for each vertex v in U , there exists a vertex u in X covers v , i.e., $\{u, v\} \in E$.

- In the k -CLIQUE problem, our goal is to determine if there is a clique of size k in the given graph G .
- In the k -CNF-SAT problem, we are given a propositional formula φ in which every clause contains at most k literals and the goal is to decide whether φ is satisfiable.

We say a set cover instance $I = (S, U, E)$ has *set cover number* m if the size of a minimum set $X \subseteq S$ such that X could cover U is m . Similarly, we say a graph G has *dominating number* m if the size of a minimum dominating set of G is m .

Circuit Complexity. For $n \in \mathbb{N}$, an n -input, m -output Boolean circuit C is a directed acyclic graph with n vertices with no incoming edges and m vertices with no outgoing edges. All nonsource vertices are called gates and are labeled with one of \vee, \wedge and \neg . The size of C , denoted by $|C|$, is the number of vertices in it. The depth of C is the length of the longest directed path from an input node to the output node. We often tacitly identify C with the function $C : \{0, 1\}^n \rightarrow \{0, 1\}^m$ it computes.

All the circuits considered in this paper are *non-uniform* and with unbounded fan-in \wedge, \vee gates unless otherwise stated.

The classes of AC^0 , $para-AC^0$ and NC^1 are defined as follows.

- AC^0 is the class of problems which can be computed by constant-depth circuit families $(C_n)_{n \in \mathbb{N}}$ where every C_n has size $\text{poly}(n)$, and whose gates have unbounded fan-in.
- $para-AC^0$ is the class of parameterized problems which can be computed by a circuit family $(C_{n,k})_{n,k \in \mathbb{N}}$ satisfying that there exist $d \in \mathbb{N}$ and a computable function f such that for every $n \in \mathbb{N}, k \in \mathbb{N}$, $C_{n,k}$ has depth d and size $f(k)\text{poly}(n)$, and whose gates have unbounded fan-in.
- NC^1 is the class of problems which can be computed by a circuit family $(C_n)_{n \in \mathbb{N}}$ where C_n has depth $O(\log n)$ and size $\text{poly}(n)$, and whose gates have fan-in 2.

Covering Arrays. A covering array $CA(N; t, p, v)$ is an $N \times p$ array A whose cells take values from a set V of size v and the set of rows of every $N \times t$ subarray of A is the whole set V^t . The smallest number N such that $CA(N; t, p, v)$ exists is denoted by $CAN(t, p, v)$. Covering arrays are discussed extensively since 1990s for they play an important role in interaction testing in complex engineered systems. The recent discussion about the upper bounds of the size of covering arrays could be found in [21].

In this article, we always assume $V = \{0, 1\}$. It is noted that in [13], a covering array $CA(N; k, n, 2)$ is also called an (n, k) -universal set.

Due to space limitations for some proofs we refer to the full version of the paper. Logarithms have base 2. Fractions and irrational numbers are rounded up if necessary.

3 Introducing Gap to k -SetCover Problem

In this section, we use the gap-gadget presented in [13], which introduces a gap for k -SETCOVER problem. Lemma 1 gives an upper bound for $CAN(k, n, 2)$. The next two lemmas also follow the idea from Lin’s work [13]. Lemma 2 allows us to construct gap gadgets with $h \leq \frac{\log n}{\log \log n}$ and $k \log \log n \leq \log n$. In Lemma 3, we present the construction which introduces gaps to set cover instances.

Definition 1. A (k, n, m, ℓ, h) -Gap-Gadget is a bipartite graph $T = (A, B, E)$ satisfying the following conditions.

- (G1) A is partitioned into (A_1, \dots, A_m) where $|A_i| = \ell$ for every $i \in [m]$.
- (G2) B is partitioned into (B_1, \dots, B_k) where $|B_j| = n$ for every $j \in [k]$.
- (G3) For each $b_1 \in B_1, \dots, b_k \in B_k$, there exists $a_1 \in A_1, \dots, a_m \in A_m$ such that a_i is adjacent to b_j for $i \in [m], j \in [k]$.
- (G4) For any $X \subseteq B$ and $a_1 \in A_1, \dots, a_m \in A_m$, if a_i has $k + 1$ neighbors in X for $i \in [m]$, then $|X| > h$.

Lemma 1. $CAN(k, n, 2) \leq k2^k \log n$ for $n \geq 5$.

Lemma 2. There is a constant-depth circuit family $(C_{k,n,h})_{k,n,h \in \mathbb{N}}$ which, for sufficiently large n satisfying $\log n \log \log \log n \geq 3(\log \log n)^2$ and $k, h \in \mathbb{N}$ with $h \leq \frac{\log n}{\log \log n}$ and $k \log \log n \leq \log n$, given $S = S_1 \cup \dots \cup S_k$ with $|S_i| = n$ for $i \in [k]$, outputs a $(k, n, n \log h, h^k, h)$ -Gap-Gadget. Furthermore, $C_{k,n,h}$ has size $O(k^2 h^{2k+1} n^2)$.

Proof. Let $m = n \log h$. Note that $\log((h \log h)2^{h \log h} \log m) \leq \log h + \log \log h + h \log h + \log \log \log h + \log \log n \leq (h+2) \log h + \log \log n \leq \log n - (\frac{\log n \log \log \log n}{\log \log n} - 3 \log \log n) \leq \log n$, that is, $(h \log h)2^{h \log h} \log m \leq n \leq n \log h$; by Lemma 1, we know that there exists a covering array $CA(n \log h; k, n, 2)$, denoted by S .

We partition every row of S into $n = \frac{m}{\log h}$ blocks so that each block has length $\log h$, interpreted as an integer in $[h]$. From the $m \times n$ numbers of S , we could obtain an $m \times n$ matrix M by setting $M_{r,c}$ to be the c -th integer of the r -th row.

Claim. For any $C \subseteq [n]$ with $|C| \leq h$, there exists $r \in [m]$ such that $|\{M_{r,c} \mid c \in C\}| = |C|$.

This claim says that for any $C = \{i_1, \dots, i_j\} \subseteq [n]$ for $j \leq h$, there is a row r such that the i_1 -th, \dots , i_j -th numbers of r are distinct. This is because we could choose the corresponding bits $C' = \cup_{c \in C} [c \log h] \setminus [(c-1) \log h]$ (since for each $c \in [n]$, the c -th number of a row is from the $((c-1) \log h + 1)$ -th bit to the $(c \log n)$ -th bit), with $|C'| \leq h \log h$; by the property of a $CA(n \log h; k, n, 2)$ covering array, there must be a row r such that for each $i_{j'} \in C$, $M_{r,i_{j'}} = j'$.

Now we construct a bipartite graph $T = (A, B, E)$ as follows.

- $A = \cup_{i \in [m]} A_i$ with each $A_i = \{\mathbf{a} \mid \mathbf{a} = (a_1, \dots, a_k), a_j \in [h] \text{ for } j \in [k]\}$;

- $B = \cup_{i \in [k]} B_i$ with $B_i = S_i$ for $i \in [k]$;
- $E = \{\{\mathbf{a}, b\} \mid \mathbf{a} \in A_i, b \in B_j \text{ and } M_{i,b} = \mathbf{a}[j], \text{ for } i \in [m], j \in [k]\}$, that is, for every $i \in [m], j \in [k]$ and every $\mathbf{a} \in A_i, b \in B_j$, if $M_{i,b} = \mathbf{a}[j]$ then we add an edge between \mathbf{a} and b .

We prove that T is a $(k, n, n \log h, h^k, h)$ -Gap-Gadget. It is clear that (G1) and (G2) hold for T . For (G3), given any $b_1 \in B_1, \dots, b_k \in B_k$, we could know that for each $i \in [m]$, $(M_{i,b_1}, \dots, M_{i,b_k}) \in A_i$, which is adjacent to b_1, \dots, b_k .

If T does not satisfy (G4), then there exists $X \subseteq B$ with $|X| \leq h$ such that there is $\mathbf{a}_1 \in A_1, \dots, \mathbf{a}_m \in A_m$ and \mathbf{a}_i has at least $k + 1$ neighbors in X for each $i \in [m]$. Since $|X| \leq h$, we could know that there is a row $r \in [m]$ such that $|\{M_{r,c} \mid c \in X\}| = X$. For this r , there exist some $j \in [k]$ such that \mathbf{a}_r has at least 2 neighbors $b_1 \neq b_2$ in B_j . However, $\{\mathbf{a}_r, b_1\}$ and $\{\mathbf{a}_r, b_2\} \in E$ means that $b_1 = b_2 = M_{i,b_1}$. This implies $|\{M_{r,c} \mid c \in X\}| < X$, which is a contradiction.

The $C_{k,n,h}$ outputs T with $\binom{kn+h^k n \log h}{2}$ bits with each bit

$$\text{bit}_T\{\mathbf{a}, b\} = \begin{cases} 1 & \text{if } M_{i,b} = \mathbf{a}[j] \\ 0 & \text{otherwise} \end{cases}$$

for $\mathbf{a} \in A_i, b \in B_j$. Hence, $C_{k,n,h}$ is with each output gate of constant depth and of size $O(k^2 h^{2k+1} n^2)$. \square

Given a set cover instance $I = (S, U, E)$, we construct the gap gadget $T = (A, B, E_T)$ with $B = S$. To use the gap gadget, we construct a new set cover instance $I' = (S', U', E')$ with $S' = S$ such that for every $X \subseteq S'$ which covers U' , there exists $a_1 \in A_1, \dots, a_m \in A_m$ witnessing that there is an $X' \subseteq X$ which covers U and each vertex of which is adjacent to a_i for some $i \in [m]$.

In the following lemma, the hypercube set system firstly presented in Feige's work [8] and is also used in [3, 13, 20]. The set $X^Y = \{f : Y \rightarrow X\}$ is considered to be all the functions from Y to X with $|X^Y| = |X|^{|Y|}$.

Lemma 3. *There is a constant-depth circuit family $(C_{n,k})_{n,k \in \mathbb{N}}$ which, for each $k \in \mathbb{N}$, given a set cover instance $I = (S, U, E)$ where $S = S_1 \cup \dots \cup S_k$ and $|S_i| = n$ for $i \in [k]$ and a (k, n, m, ℓ, h) -Gap-Gadget constructed with S as Lemma 2, outputs a set cover instance $I' = (S', U', E')$ with $S' = S$ and $U' = m|U|^\ell$ such that*

- if there exist $s_1 \in S_1, \dots, s_k \in S_k$ that can cover U , then the set cover number of I' is at most k ;
- if the set cover number of I is larger than k , then the set cover number of I' is greater than h .

Furthermore, the circuit $C_{n,k}$ has size at most $\ell(kn + m|U|^\ell)^2$.

Proof. Let $T = (A, B, E_T)$ be the (k, n, m, ℓ, h) -Gap-Gadget with $B_i = S_i$ for $i \in [k]$. $I' = (S', U', E')$ is defined as follows.

- $S' = S$;
- $U' = \cup_{i \in [m]} U^{A_i}$;
- For every $s \in S'$ and $f \in U^{A_i}$ for each $i \in [m]$, $\{s, f\} \in E'$ if there is an $a \in A_i$ such that $\{s, f(a)\} \in E$ and $\{a, s\} \in E_T$.

Completeness. If there exist $s_1 \in S_1, \dots, s_k \in S_k$ that can cover U , then we show that for each $f \in U'$, it is covered by some vertex in $C = \{s_1, \dots, s_k\}$. Suppose $f \in U^{A_i}$. By (G3) we could know that, for $s_1 \in S_1, \dots, s_k \in S_k$, there exists $a_1 \in A_1, \dots, a_m \in A_m$ such that a_p is adjacent to s_q for $p \in [m], q \in [k]$. Since C covers U , there must be $s_j \in C$ for some $j \in [k]$ covers $f(a_i)$. That is, we have $\{f(a_i), s_j\} \in E$ and $\{a_i, s_j\} \in E_T$, which means s_j covers f .

Soundness. If the set cover number of I is greater than k , we show that for every $X \subseteq S'$ that covers U' , we must have $|X| > h$.

Claim. For any $X \subseteq S'$ that covers U' , there exist $a_1 \in A_1, \dots, a_m \in A_m$ that $|N_T(a_i) \cap X| \geq k + 1$ for every $i \in [m]$.

Otherwise, there is some $i \in [m]$ such that for any $a \in A_i$, we have $|N_T(a) \cap X| \leq k$, which means there is some $u \in U$ not covered by $N_T(a) \cap X$ since the covering number of I is greater than k . For $f \in U^{A_i}$ such that $f(a') = u$ for any $a' \in A_i$, it is covered by S only if it is covered by some $s \in S \setminus N_T(a)$ since u can only be covered by $S \setminus N_T(a)$. However, for any $s \in S \setminus N_T(a)$, s is not a neighbor of a . That is, f is not adjacent to $S \setminus N_T(a)$, either. Hence, f is not covered by X , a contradiction.

With the claim, we know that for any $X \subseteq S'$ that covers U' , there exist $a_1 \in A_1, \dots, a_m \in A_m$ that a_i has $k + 1$ neighbors in X for every $i \in [m]$. With (G4), we must have $|X| \geq h$.

The $C_{n,k}$ outputs I' with $\binom{kn+m}{2}^{|U|^\ell}$ bits with each bit

$$\text{bit}_{I'}\{s, f\} = \bigvee_{a \in A_i} \text{bit}_T\{s, a\} \wedge \text{bit}_I\{s, f(a)\}$$

for $s \in S', f \in U^{A_i}$. Hence, $C_{n,k}$ is with each output gate of depth at most 6 and of size at most $\ell(kn + m)|U|^\ell$. \square

4 Inapproximability of k -DominatingSet

In this section, we show the inapproximability of the dominating set problem by proving Theorems 3 and 4. To show Theorem 3, we have Lemma 4 and Lemma 5. Lemma 4 follows the idea in [13,20], presenting the circuits output a $\binom{k}{2}$ -SETCOVER instance when given a k -CLIQUE instance as input. With Lemma 4, Lemma 5 introduces circuits reducing k -CLIQUE instances to set cover instances with gaps. To prove Theorem 4, Lemma 6 is used to prove the inapproximability of set cover problem using constant-depth $n^{o(k)}$ circuits, assuming Hypothesis 1. At the end of this section, we show that Hypothesis 1 may be hard to settle by showing that it implies $NP \not\subseteq NC^1$, which remains open for decades.

Lemma 4. *There is a $(C_{n,k})_{n,k \in \mathbb{N}}$ circuit family which, given a k -CLIQUE instance G with $|V_G| = n$, outputs a set cover instance $I = (S, U, E)$ with $|U| \leq k^3 \log n$ and $S \leq \binom{k}{2} \binom{n}{2}$ such that G contains a k -clique if and only if the set cover number of I is at most $\binom{k}{2}$. Furthermore, $C_{n,k}$ has constant depth and size at most $\binom{kn+k^3 \log n}{2}$.*

Lemma 5. *There is a $(C_{n,k})_{n,k \in \mathbb{N}}$ circuit family which, given a k -CLIQUE instance G with $|V_G| = n$ and $k < 2\sqrt[3]{\log n}$, outputs a set cover instance $I = (S, U, E)$ with $|U| \leq n^6$ and $S \leq \binom{n}{2}$ such that*

- if G contains a k -clique, then the set cover number of I is at most $\binom{k}{2}$;
- if G contains no k -clique, then the set cover number of I is greater than $(\frac{\log n}{\log \log n})^{1/\binom{k}{2}}$.

Furthermore, $C_{n,k}$ has constant depth and size $O(n^{12})$.

Theorem 3. *Given a set cover instance $I = (S, U, E)$ with $n = |S| + |U|$, for $48 < k < 2\sqrt[3]{\log n}$, any constant-depth circuit of size $O(n^{\frac{\sqrt{k}}{48}})$ cannot distinguish between*

- the set cover number of I is at most k , or
- the set cover number of I is greater than $(\frac{\log n}{\log \log n})^{1/\binom{k}{2}}$.

Proof. In [19], Rossman showed that for every $k \in \mathbb{N}$, the k -CLIQUE problem on n -vertex graphs require constant-depth circuits of size $\omega(n^{\frac{k}{4}})$. Now if there is a constant-depth circuit $C_{n,k}$ of size $O(n^{\frac{\sqrt{k}}{48}})$ could distinguish between the set cover number of I is at most k or greater than $(\frac{\log n}{\log \log n})^{1/\binom{k}{2}}$, then by Lemma 5, given $k \in \mathbb{N}$ and a graph G with $|V_G| = n$, we could construct a set cover instance I with vertex number $O(n^6)$ satisfying that if G has a k -clique then the set cover number of I is at most $\binom{k}{2}$ and otherwise it is greater than $(\frac{\log n}{\log \log n})^{1/\binom{k}{2}}$ —we could use $C_{(n^6), \binom{k}{2}}$ to decide the set cover number of I is either $\leq \binom{k}{2}$ or $> (\frac{\log n}{\log \log n})^{1/\binom{k}{2}}$. The circuits are of size $O(\binom{n^6}{2}^{\sqrt{\binom{k}{2}}/48}) + O(n^{12}) = O(n^{\frac{k}{4}})$ when $k > 48$, which contradicts the result in [19]. \square

Note that Theorem 3 implies Theorem 1 since for every set cover instance $I = (S, U, E)$ we could construct a dominating set instance $I' = (S \cup U, E \cup \{\{u, v\} | u, v \in S\})$ simply by adding edges to S so that it becomes a clique. Then the dominating number of I' is the same as the set cover number of I .

Next we show that the inapproximability of the set cover problem for constant-depth circuits of size $n^{o(k)}$ assuming Hypothesis 1.

Lemma 6. *There is a circuit family $(C_{n,k})_{n,k \in \mathbb{N}}$ which for every $k \in \mathbb{N}$, given a CNF-SAT instance φ with n variables and Cn clauses where n is much larger than k and C , outputs $N \leq 2^{\frac{5n}{2k}}$ a set cover instance $I = (S, U, E)$ satisfying*

- $|S| + |U| \leq N$;
- if φ is satisfiable, then the set cover number is at most k ;
- if φ is not satisfiable, then the set cover number is greater than $\sqrt[k]{\frac{\log N}{3 \log \log N}}$;

Furthermore, $C_{n,k}$ has constant depth and size at most $2^{\frac{5n}{2k}}$.

Theorem 4. *Assuming Hypothesis 1, there is $\varepsilon > 0$ such that, given a set cover instance $I = (S, U, E)$ with $n = |S| + |U|$, any constant-depth Boolean circuit of size $n^{\varepsilon k}$ cannot distinguish between*

- the set cover number of I is at most k , or
- the set cover number of I is greater than $\sqrt[k]{\frac{\log n}{3 \log \log n}}$.

Proof. By Hypothesis 1, there exists $\delta > 0$ such that no constant-depth circuits of size $2^{\delta n}$ can decide whether the 3-CNF-SAT instance φ is satisfiable where n is the number of variables of φ . For every 3-CNF-SAT formula φ , there is a constant-depth circuit $C_{n,k}$ of size $2^{\frac{5n}{2k}}$ which, given φ , computes a set cover instance $I = (S, U, E)$ with $|S| + |U| \leq N$ for $N \leq 2^{\frac{5n}{2k}}$ whose set cover number is either at most k or greater than $\sqrt[k]{\frac{\log N}{3 \log \log N}}$ by Lemma 4. Now take $\varepsilon = \delta/6$.

If constant-depth circuit C_n of size $n^{\varepsilon k}$ could distinguish between the set cover number of I is either at most k or greater than $\sqrt[k]{\frac{\log n}{3 \log \log n}}$ where n is the vertex number of the given set cover instance I , then we could use $C_{\binom{N}{2}}$ to determine the set cover number of I is whether at most k , i.e., to decide if φ is satisfiable. The used circuits have size at most $2^{\frac{5n}{2k}} + (N^2)^{\varepsilon k} \leq 2^{6\varepsilon n} = 2^{\delta n}$, which contradicts Hypothesis 1. □

Using the same trick for Theorem 1, we could know that Theorem 4 implies Theorem 2.

Though Hypothesis 1 is much weaker than ETH, we find that it is still very hard to settle by showing the following theorem.

Lemma 7. *For every $L \in \text{NC}^1$, i.e., there exists $c \in \mathbb{N}$ such that L could be computed by a family of circuits $(C_n)_{n \in \mathbb{N}}$ satisfying C_n has size at most n^c and depth at most $c \log n$, there exists $d \in \mathbb{N}$ such that there is a family of circuits $(C'_n)_{n \in \mathbb{N}}$ which satisfies*

- $s \in L$ if and only if $C_{|s|} = 1$;
- C_n has depth d and size $n^{3c/2}(2^{n^{2c/d}+1} + 1)$.

Theorem 5. *Hypothesis 1 implies $\text{NP} \not\subseteq \text{NC}^1$.*

Proof. We show that Hypothesis 1 implies 3-CNF-SAT $\notin \text{NC}^1$. If there exists $c \in \mathbb{N}$ such that 3-CNF-SAT could be computed by a family of circuits $(C_n)_{n \in \mathbb{N}}$ satisfying C_n has size at most n^c and depth at most $c \log n$. By Lemma 7, 3-CNF-SAT could be computed by $5c$ -depth, size $n^{3c/2}(2^{n^{2/5}+1} + 1) = O(2^{\sqrt{n}})$ circuits for large n , which contradicts Hypothesis 1. □

5 Conclusions

We have presented that para-AC⁰ circuits could not approximate the k -DOMSET problem with ratio $f(k)$ for any computable function f . With the hypothesis that

3-CNF-SAT problem cannot be computed by constant-depth circuits of size $2^{\delta n}$ for some $\delta > 0$, we could show that constant-depth circuits of size $n^{o(k)}$ cannot distinguish graphs whose dominating numbers are either $\leq k$ or $> \sqrt[k]{\frac{\log n}{3 \log \log n}}$.

A natural question is to settle the hypothesis, which may be hard since we show that it implies $\text{NP} \not\subseteq \text{NC}^1$. Another question is to ask: Are constant-depth circuits of size $n^{o(k)}$ unable to distinguish graphs whose dominating numbers are either $\leq k$ or $> \sqrt[k]{\frac{\log n}{3 \log \log n}}$ *without assuming Hypothesis 1*?

Acknowledgement. I am grateful to Yijia Chen, Bundit Laekhanukit and Chao Liao for many helpful discussions and valuable comments. I also thank the anonymous referees for their detailed comments. This research is supported by National Natural Science Foundation of China (Project 61872092).

References

1. Ajtai, M.: Σ_1^1 -formulae on finite structures. *Ann. Pure Appl. Logic* **24**(1), 1–48 (1983). [https://doi.org/10.1016/0168-0072\(83\)90038-6](https://doi.org/10.1016/0168-0072(83)90038-6)
2. Alon, N., Moshkovitz, D., Safra, S.: Algorithmic construction of sets for k -restrictions. *ACM Trans. Algorithms* **2**(2), 153–177 (2006). <https://doi.org/10.1145/1150334.1150336>
3. Chalermsook, P., et al.: From gap-ETH to FPT-inapproximability: clique, dominating set, and more. In: 2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS), pp. 743–754. IEEE, Berkeley, October 2017. <https://doi.org/10.1109/FOCS.2017.74>
4. Chen, Y., Flum, J.: Some lower bounds in parameterized ac⁰. In: Faliszewski, P., Muscholl, A., Niedermeier, R. (eds.) 41st International Symposium on Mathematical Foundations of Computer Science (MFCS 2016). *Leibniz International Proceedings in Informatics (LIPIcs)*, vol. 58, pp. 27:1–27:14. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl (2016). <https://doi.org/10.4230/LIPIcs.MFCS.2016.27>
5. Chvatal, V.: A greedy heuristic for the set-covering problem. *Math. Oper. Res.* **4**(3), 233–235 (1979). <https://doi.org/10.1287/moor.4.3.233>
6. Dinur, I., Steurer, D.: Analytical approach to parallel repetition. In: *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing, STOC 2014*, pp. 624–633. ACM, New York (2014). <https://doi.org/10.1145/2591796.2591884>
7. Elberfeld, M., Stockhusen, C., Tantau, T.: On the space and circuit complexity of parameterized problems: classes and completeness. *Algorithmica* **71**(3), 661–701 (2015). <https://doi.org/10.1007/s00453-014-9944-y>
8. Feige, U.: A threshold of $\ln n$ for approximating set cover. *J. ACM* **45**(4), 634–652 (1998). <https://doi.org/10.1145/285055.285059>
9. Furst, M., Saxe, J.B., Sipser, M.: Parity, circuits, and the polynomial-time hierarchy. *Math. Syst. Theor.* **17**(1), 13–27 (1984). <https://doi.org/10.1007/BF01744431>
10. Håstad, J.: Almost optimal lower bounds for small depth circuits. In: *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing, STOC 1986*, pp. 6–20. ACM, New York (1986). <https://doi.org/10.1145/12130.12132>
11. Johnson, D.S.: Approximation algorithms for combinatorial problems. *J. Comput. Syst. Sci.* **9**(3), 256–278 (1974). [https://doi.org/10.1016/S0022-0000\(74\)80044-9](https://doi.org/10.1016/S0022-0000(74)80044-9)

12. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W., Bohlinger, J.D. (eds.) *Complexity of Computer Computations. The IBM Research Symposia Series*, pp. 85–103. Springer, Boston (1972). <https://doi.org/10.1007/978-1-4684-2001-2>
13. Lin, B.: A Simple Gap-producing Reduction for the Parameterized Set Cover Problem (2018). <https://sites.google.com/site/bingkai314159/gapsetcover.pdf>
14. Lovász, L.: On the ratio of optimal integral and fractional covers. *Discrete Math.* **13**(4), 383–390 (1975). [https://doi.org/10.1016/0012-365X\(75\)90058-8](https://doi.org/10.1016/0012-365X(75)90058-8)
15. Lund, C., Yannakakis, M., Yannakakis, M.: On the hardness of approximating minimization problems. *J. ACM* **41**(5), 960–981 (1994). <https://doi.org/10.1145/185675.306789>
16. Moshkovitz, Dana: The projection games conjecture and the NP-hardness of in n -approximating set-cover. In: Gupta, Anupam, Jansen, Klaus, Rolim, José, Servedio, Rocco (eds.) *APPROX/RANDOM-2012. LNCS*, vol. 7408, pp. 276–287. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32512-0_24
17. Murray, C., Williams, R.: Circuit lower bounds for nondeterministic quasipolytime: an easy witness lemma for NP and NQP. In: *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018*, pp. 890–901. ACM, New York (2018). <https://doi.org/10.1145/3188745.3188910>
18. Raz, R., Safra, S.: A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In: *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing, STOC 1997*, pp. 475–484. ACM, New York (1997). <https://doi.org/10.1145/258533.258641>
19. Rossman, B.: On the constant-depth complexity of k -clique. In: *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing, STOC 2008*, pp. 721–730. ACM, New York (2008). <https://doi.org/10.1145/1374376.1374480>
20. Karthik, C.S., Laekhanukit, B., Manurangsi, P.: On the parameterized complexity of approximating dominating set. In: *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing - STOC 2018*, pp. 1283–1296. ACM Press, Los Angeles (2018). <https://doi.org/10.1145/3188745.3188896>
21. Sarkar, K., Colbourn, C.J.: Upper bounds on the size of covering arrays. *SIAM J. Discrete Math.* **31**(2), 1277–1293 (2017). <https://doi.org/10.1137/16M1067767>
22. Slavík, P.: A tight analysis of the greedy algorithm for set cover. *J. Algorithms* **25**(2), 237–254 (1997). <https://doi.org/10.1006/jagm.1997.0887>
23. Stein, S.K.: Two combinatorial covering theorems. *J. Comb. Theor. Ser. A* **16**(3), 391–397 (1974). [https://doi.org/10.1016/0097-3165\(74\)90062-4](https://doi.org/10.1016/0097-3165(74)90062-4)
24. Williams, R.: New algorithms and lower bounds for circuits with linear threshold gates. In: *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing, STOC 2014*, pp. 194–202. ACM, New York (2014). <https://doi.org/10.1145/2591796.2591858>
25. Williams, R.: Nonuniform ACC circuit lower bounds. *J. ACM* **61**(1), 2:1–2:32 (2014). <https://doi.org/10.1145/2559903>



Mutual Visibility by Robots with Persistent Memory

Subhash Bhagat and Krishnendu Mukhopadhyaya^(✉)

Advanced Computing and Microelectronics Unit, Indian Statistical Institute,
Kolkata, India

subhash.bhagat.math@gmail.com, krishnendu@isical.ac.in

Abstract. This paper addresses one of the fundamental geometric formation problems, namely the *mutual visibility* problem, for a set of semi-synchronous, opaque robots occupying distinct positions in the Euclidean plane. Since robots are opaque, if three robots lie on a line, the middle robot obstructs the visions of the two other robots. The mutual visibility problem requires the robots to coordinate their movements to form a configuration, within finite time and without collision, in which no three robots are collinear. We assume that robots are endowed with constant bits of persistent memory. We consider the FSTATE computational model [4] in which the persistent memory is used by the robots only to remember their previous internal states. This piece of information is not communicated or visible to the other robots. Except from this persistent memory, robots are oblivious i.e., they do not carry forward any other information from their previous computational cycles. The paper presents a distributed algorithm to solve the mutual visibility problem for a set of semi-synchronous robots using only 1 bit of persistent memory. The proposed algorithm also provides a self-stabilizing solution to the problem. The algorithm does not impose any other restriction on the capability of the robots and guarantees collision-free movements for the robots.

Keywords: Swarm robots · Mutual visibility problem · Semi-synchronous · Persistent memory · Self-stabilizing

1 Introduction

A *swarm* of robots is a multi-robot system consisting of autonomous, homogeneous, small mobile robots which are capable of carrying out some task in a cooperative environment. The robots are modelled as points on the two-dimensional plane. The robots are indistinguishable by their appearances. All of them have identical capabilities and execute same algorithm. They do not share a global coordinate system; each robot has its own local coordinate system. The directions and orientations of the local coordinate axes may vary. Each robot executes the computational cycles consisting of three phases *Look-Compute-Move*. In *Look*

phase, a robot takes the snapshot of its surroundings and maps the locations of the other robots w.r.t. its local coordinate system. In *Compute* phase, a robot uses the information gathered in the *Look* phase to compute a destination point. In *Move* phase, it moves towards the computed destination point.

In persistent memory model, robots are endowed with constant amount of persistent memory (the robots are otherwise oblivious) [1]. This persistent memory can be used in three different ways: (i) the robots can set limited communications between themselves using *visible lights* which can assume a constant number of predefined colors to represent their different states and also to retain some constant amount of information about their previous states or (ii) only to remember information about their last states (FSTATE model) or (iii) the robots can use visible lights only to communicate with other robots in the system and they do not remember the colors of the lights of their last computational cycle (FCOMM model) [4]. Thus, the persistent memory can be used for communication or for internal memory or for both. In this work, robots use persistent memory only for internal memory.

The *mutual visibility* problem is defined as follows: for a set of robots initially occupying distinct positions in the two dimensional plane, the mutual visibility problem asks the robots to form a configuration, within finite time and without collision, in which no three robots are collinear.

1.1 Earlier Works

The mutual visibility problem was first studied by Di Luna et al. [13]. They presented a distributed algorithm to solve the problem for a set of semi-synchronous oblivious robots. Their approach assumes that the robots have the knowledge of total number of robots in the system. Later, their algorithm was analysed and modified by Sharma et al. [11] to improve the round complexity of the algorithm for fully synchronous robots. Di Luna et al. [12] were the first to study the mutual visibility problem for the robots with persistent memory. They solved the problem for the semi-synchronous robots with 3 colors and for asynchronous robots with 3 colors under one axis agreement. Later, Sharma et al. [10] proved that the mutual visibility problem is solvable using only 2 colors for semi-synchronous robots and using 2 colors for asynchronous robots under one-axis agreement. Sharma et al. [8] proposed a solution to the problem which runs in constant time for a set of asynchronous robots using 47 colors. Bhagat and Mukhopadhyay [9] solved the problem for a set of asynchronous robots using 7 colors. In their solution, each robot moves exactly once. The mutual visibility problem has also been considered under different fault models [5, 6, 14] and also for *fat robots* [7].

The only solution to the mutual visibility problem for oblivious asynchronous robots has been proposed in [2] under the assumption that the robots have an agreement in one coordinate axis and knowledge of total number of robots in the system. Thus, all the existing algorithms for the mutual visibility problem assumes either persistent memory for both communication and internal memory purpose or axis agreement or the knowledge of total number of robots.

1.2 Our Contribution

This paper studies the *mutual visibility* problem for a set of n semi-synchronous robots in the Euclidean plane. A simple but elegant distributed algorithm has been proposed to solve the problem for a set of robots endowed with a constant amount of persistent memory. The proposed algorithm considers FSTATE model which does not have communication overhead of FCOMM model. The persistent memory is used only to remember information about their previous states. The proposed algorithm does not assume any other extra assumptions like agreement on the coordinate axes or chirality, knowledge of n , rigidity of movements. In spite of these weak assumptions, it is shown that the mutual visibility problem is solvable for a set of semi-synchronous robots using only 1 bit of persistent internal memory. The contribution of this paper has following significance.

- While all the existing solutions of the mutual visibility problem for semi-synchronous robots have considered either knowledge of n or persistent memory for both communication and internal memory purposes (combination of FSTATE and FCOMM model), our approach assumes FSTATE model without knowledge of n (this makes system easily scalable).
- In all the existing solutions for the mutual visibility problem, the convex hull of the initial positions of the robots does not remain invariant. The solution of this work maintains the convex hull of the initial robot positions if all the robots initially do not lie on a single line. Furthermore, in all the works with persistent memory, the robots move even if the robots are completely visible to each other. In our algorithm, if the robots are completely visible to each robot, they do not move.
- In our approach, not all robots move. Only the robots which block the vision of the other robots move. Again, the distances they traverse during their movements are kept as small as needed. These help to provide an energy efficient solution to the problem.
- The proposed algorithm is self-stabilizing. Even if robots start with different states, the algorithm achieves its final goal.
- The solution also provides collision free movements for the robots.
- To the best of our knowledge, this paper is the first attempt to study the mutual visibility problem under FSTATE model.

2 Assumptions and Notations

This paper considers a set of n semi-synchronous point robots in the Euclidean plane. The robots are opaque. However, the visibility range of a robot is unlimited. The robots have no knowledge about the total number of robots in the system. The movements of the robots are *non-rigid*. The robots do not have any explicit communication power. Each robot has 1 bit of internal persistent memory. The 1 bit memory stores information about predefined specific states of the robot. This internal bit does not change automatically and it is persistent. Let $s_i(t)$ be the binary variable which denotes the value stored in the internal

memory of the robot r_i at time $t \in \mathbb{N}$. Except for this persistent memory, the robots are oblivious i.e., they do not remember any other data of their previous computational cycles. Initially all the robots occupy distinct locations and they are stationary.

- **Configurations of the robots:** Let $\mathcal{R} = \{r_1, r_2, \dots, r_n\}$ denote the set of n robots. The position of robot r_i at time t is denoted by $r_i(t)$. A configuration of robots, $\mathcal{R}(t) = \{r_1(t), \dots, r_n(t)\}$, is the set of positions occupied by the robots at time t . $\tilde{\mathcal{C}}$ is the set of all such robot configurations. We partition $\tilde{\mathcal{C}}$ into two classes: $\tilde{\mathcal{C}}_L$ and $\tilde{\mathcal{C}}_{NL}$, where $\tilde{\mathcal{C}}_L$ is the collection of configurations in which all the robots in \mathcal{R} lie on a straight line and $\tilde{\mathcal{C}}_{NL}$ consists of configurations in which there exist at least three non-collinear robot positions occupied by the robots in \mathcal{R} . We say that a robot configuration $\mathcal{R}(t)$ is in *general position* if no three robot positions in $\mathcal{R}(t)$ are collinear. By $\tilde{\mathcal{C}}_{GP}$, we denote the set of all configurations of \mathcal{R} which are in general position. Clearly $\tilde{\mathcal{C}}_{GP} \subset \tilde{\mathcal{C}}_{NL}$.
- **Measurement of angles:** By an angle between two line segments, if not stated otherwise, we mean the angle which is less than or equal to π .
- **Vision of a robot:** If three robots r_i, r_j and r_k are collinear with r_j lying in between r_i and r_k , then r_i and r_k are not visible to each other. We define the vision, $\mathcal{V}_i(t)$, of robot r_i at time t to be the set of robot positions visible to r_i (excluding r_i). The *visibility polygon* of r_i at time t , denoted by $STR(r_i(t))$, is defined as follows: sort the points in $\mathcal{V}_i(t)$ angularly in anti clockwise direction w.r.t. $r_i(t)$, starting from any robot position in $\mathcal{V}_i(t)$. Then connect them in that order to generate the polygon $STR(r_i(t))$.
- A straight line \mathcal{L} is called a *line of collinearity* if it contains more than two distinct robot positions. A robot occupying a position on \mathcal{L} is termed a *collinear* robot. For a robot r_i , let $\mathcal{B}_i(t)$ denote the set of all lines of collinearity on which r_i is a collinear robot at time $t \in \mathbb{N}$.
- Consider a line of collinearity \mathcal{L} at time t . A robot r_i on \mathcal{L} is called a *non-terminal* robot if $r_i(t)$ is a point between two other robot positions on \mathcal{L} . A robot which is not a non-terminal robot is called a *terminal* robot.
- A non-terminal robot position $r_i(t)$ on a line of collinearity \mathcal{L} is called a *junction robot position* if there is another line of collinearity \mathcal{L}_1 such that $r_i(t)$ lies at the intersection point between \mathcal{L} and \mathcal{L}_1 .
- By \overline{pq} , we denote the closed line segment joining two points p and q , including the end points p and q . Let (p, q) denote the open line segment joining the points p and q , excluding the two end points p and q . Let $|\overline{pq}|$ denote the length of \overline{pq} .
- $\mathbf{d}_{ij}^k(\mathbf{t})$: Let $\mathcal{L}_{ij}(t)$ denote the straight line joining $r_i(t)$ and $r_j(t)$. The perpendicular distance of the line $\mathcal{L}_{ij}(t)$ from the point $r_k(t)$ is denoted by $d_{ij}^k(t)$.
- $\mathbf{D}_i(\mathbf{t})$: $D_i(t)$ is the minimum distance of any two robot positions in $\{r_i(t)\} \cup \mathcal{V}_i(t)$.

3 Algorithm for the Mutual Visibility Problem

Consider an initial configuration $\mathcal{R}(t_0)$ of robots. If $\mathcal{R}(t_0)$ contains no non-terminal robot, then $\mathcal{R}(t_0) \in \tilde{C}_{GP}$ i.e., all the robots in the system are visible to each other. On the contrary, if $\mathcal{R}(t_0)$ contains at least one non-terminal robot, then there are at least two robots which are not visible to each other. In this scenario, to achieve complete visibility, robots need to coordinate their movements. In this process one has to decide two main things; (i) which are the robots to move: terminals or non-terminals? (ii) what should be their destination point to move? First, we try to give an intuitive idea to resolve these issues and then we describe our algorithm in details.

3.1 Eligible Robots for Movements

Non-terminal robots block the vision of the other robots. In our approach, we choose non-terminal robots for movements. Since one of our main objectives is to maintain the convex hull of the initial robot positions and the robots lying at the vertices of the convex hull are terminal robots, we do not move terminal robots. A robot can easily determine whether it is a terminal robot or non-terminal robot.

3.2 Different Types of Movements

A robot uses its 1 bit of internal memory to remember the information about its last movement. It uses 0 and 1 in its persistent memory for this purpose. Initially all robots have 0 in their respective persistent memory. If the internal bit is 0, a robot moves not along any line of collinearity and this move is called a *type-0* move. If internal bit is 1, a robot moves along a line of collinearity and this move is called a *type-1* move.

3.3 States of a Robot

A robot uses its persistent 1 bit memory to remember information about its last movement. Initially all robots have 0 in their persistent memory.

- If a robot is terminal and its internal bit is 0, it is a terminal robot since the initial configuration.
- If a robot is terminal and its internal bit is 1, it was a non-terminal robot in the initial configuration and has become terminal during the execution of the algorithm.
- If a robot is non-terminal and its internal bit is 0, it is a non-terminal robot since the initial configuration and either it has made no move or has made a type-1 move.
- If a robot is non-terminal and its internal bit is 1, it is a non-terminal robot since the initial configuration and it has made a type-0 move.

3.4 Computation of Destination Point

Destination points of the robots are computed in such way that (i) they do not create new collinearities by moving to these positions and (ii) the total number of collinear robots in the system should decrease within finite number of movements. The algorithm terminates when system contains no non-terminal robot. Let r_i be an arbitrary non-terminal robot at time $t \geq t_0$. To find the new position of r_i , we first decide on the direction of movement and then the amount of displacement along this direction. While computing the new destination point of r_i , two things should be taken care of. The new position of r_i should not block the visibility of the other robots. The movements of the robots should be collision free. Depending on the current configuration $\mathcal{R}(t)$, the destination point for r_i is computed as follows.

– **Case-1:** $\mathcal{R}(t) \in \tilde{\mathbf{C}}_{\text{NL}}$

Consider the set of angles $\Gamma_i(t)$ defined as follows:

$$\Gamma_i(t) = \{\angle r_j r_i r_k : r_j, r_k \text{ are two consecutive vertices on } STR(r_i(t))\}$$

- **The direction of movement:** Let $\alpha_i(t)$ denote the angle in $\Gamma_i(t)$ having the maximum value if the maximum value is less than π , otherwise the 2^{nd} maximum value (tie, if any, is broken arbitrarily). The bisector of $\alpha_i(t)$ is denoted by $Bisec_i(t)$. It is a ray from $r_i(t)$. If persistent bit is 0, r_i makes a type-0 move and its direction of movement is along $Bisec_i(t)$. Before starting its movement, r_i changes its persistent bit to 1. It may be noted that any other suitable direction for type-0 move would work fine for robot r_i . If persistent bit is 1, r_i makes a type-1 move. r_i arbitrarily chooses a line of collinearity from $\mathcal{B}_i(t)$ and moves along this line. Before starting a type-1 move, r_i changes its persistent bit to 0.
- **The amount of displacement:** Let $d_i(t) = \text{minimum}\{d_{ij}^k(t), d_{ik}^j(t), d_{jk}^i(t) : \forall r_j, r_k \in \mathcal{V}_i(t)\}$. The amount of displacement of r_i at time t is denoted by $\sigma_i(t)$ and it is defined as follows,

$$\sigma_i(t) = \frac{U}{3^{4v_i(t)}}$$

where $U = \text{minimum}\{d_i(t), D_i(t)\}$ and $v_i(t) = |\mathcal{V}_i(t)|$.

Three non-collinear robots become collinear when the triangle formed by their positions collapses to a line. The amount $\sigma_i(t)$ is chosen to be a small fraction of $d_{ij}^k(t)$ for all $r_j(t), r_k(t) \in \mathcal{V}_i(t)$ in order to guarantee that no new collinearity is generated during the movements of the robots. Other suitable values will also work.

- **The destination point:** Let $\hat{r}_i(t)$ be the point on $Bisec_i(t)$ at distance $\sigma_i(t)$ from $r_i(t)$ if $s_i(t) = 0$. Otherwise, $\hat{r}_i(t)$ is a point on a line $\mathcal{L} \in \mathcal{B}_i(t)$ at distance $\sigma_i(t)$ from $r_i(t)$ (choose arbitrarily any one of the two directions along \mathcal{L}). The destination point of $r_i(t)$ is $\hat{r}_i(t)$.

– **Case-2:** $\mathcal{R}(t) \in \tilde{\mathcal{C}}_L$

There is only one line of collinearity, say $\hat{\mathcal{L}}$, in the system. Only two robots are terminal. Once one of them moves, the present configuration is converted into a configuration in $\tilde{\mathcal{C}}_{NL}$.

- **The direction of movement:** Let \mathcal{L}^* be the line perpendicular to $\hat{\mathcal{L}}$ at the point $r_i(t)$. The robot r_i arbitrarily chooses a direction along \mathcal{L}^* and moves along that direction. Let \mathcal{L}_d^* denote the direction of movement of r_i . Since all robots are collinear, this movement is a type-0 move. Before starting this move, r_i changes its persistent bit to 1.
- **The amount of displacement:** In this, the amount of displacement $\hat{\sigma}_i(t)$ is defined as follows:

$$\hat{\sigma}_i(t) = \frac{D_i(t)}{3^4}$$

- **The destination point:** Let $\bar{r}_i(t)$ be the point on \mathcal{L}_d^* at the distance $\hat{\sigma}_i(t)$ from $r_i(t)$. The destination point of r_i is $\bar{r}_i(t)$.

3.5 Termination

A robot terminates the execution of algorithm *MutualVisibility()* when it finds itself as a terminal robot. Thus, an initially terminal robot terminates just in one round.

Robots use the algorithm *ComputeDestination()* to compute its destination point and use algorithm *MutualVisibility()* to obtain complete visibility.

3.6 Correctness

To prove the correctness of our algorithm, we need to prove the following for any configuration: (i) three non-collinear robots in a particular round do not become collinear in any of the succeeding rounds (ii) within finite number of rounds at least one non-terminal robot becomes terminal and (iii) movements of the robots are collision free. If three non-collinear robots become collinear, then the triangle formed by their positions should collapse into either a line or a point. For arbitrary three non-collinear robots r_i , r_j and r_k , we prove that none of $d_{ij}^k(t)$, $d_{ik}^j(t)$ and $d_{jk}^i(t)$ becomes zero. Without loss of generality, we prove that $d_{ij}^k(t)$ will never vanish, during the execution of our algorithm. We estimate the maximum decrement in the value of $d_{ij}^k(t)$ in a particular round, due to the movements of the robots.

Lemma 1. *Let r_i, r_j and r_k be three arbitrary robots, which are not collinear at time $t \in \mathbb{N}$. During the rest of execution of algorithm *MutualVisibility()*, they do not become collinear.*

Proof. Maximum decrement in the value of $d_{ij}^k(t)$ occurs when all the three robots move simultaneously in a round. Thus, we suppose the three robots move at time t . Depending upon the positions of the robots, we have the following cases.

– **Case-1: r_i, r_j and r_k are mutually visible at t_0**

According to our approach, the displacement of a robot, in a single movement, is bounded above by $\frac{d_{ij}^k(t)}{3^4}$ (since $|\mathcal{V}_i(t)| \geq 1$). Since all the three robots move simultaneously in a round, the total decrement in the value of $d_{ij}^k(t)$ is bounded above by $\frac{3}{3^4}d_{ij}^k(t)$. It is easy to see that this bound also holds for all other scheduling of the actions of the robots. Thus, we have,

$$d_{ij}^k(t+1) > (1 - \frac{3}{3^4})d_{ij}^k(t) \quad (1)$$

Equation (1) implies that the triangle $\Delta_{ijk}(t)$ does not collapse into a line due to the movements of the robots. Since robots are semi-synchronous and t is arbitrary, these three robots never become collinear during the whole execution of the algorithm.

– **Case-2: r_i, r_j and r_k are not mutually visible at t_0**

In this case the triangle $\Delta_{ijk}(t)$ contains a triangle $\Delta_{xyz}(t)$ such that the robots lying at three vertices r_x, r_y and r_z are mutually visible to each other. Case-1 above implies that the triangle $\Delta_{xyz}(t)$ does not vanish during the movements of the robots and so does $\Delta_{ijk}(t)$. \square

Lemma 2. *Let r_i be an initially non-terminal robot. During the execution of algorithm $MutualVisibility()$, \exists a time $t \in \mathbb{N}$ such that r_i becomes a terminal robot at time t and it remains terminal for the rest of the execution of the algorithm.*

Proof. Let $\mathcal{L}_1(t')$ be a line of collinearity in $\mathcal{B}_i(t')$.

– **Case-1: $\mathcal{L}_1(t')$ does not contain a junction robot position**

In this case, r_i is a non-terminal robot on exactly one line. Since both the end robot positions on $\mathcal{L}_1(t')$ are terminal, it takes at most $2k - 1$ number of movements for the non-terminal robots on $\mathcal{L}_1(t')$ to become terminal, where k is number of non-terminal robots on $\mathcal{L}_1(t')$ (Fig. 1).

– **Case-2: $\mathcal{L}_1(t')$ contains a junction robot position**

We consider different possible configurations of the robot positions on the line $\mathcal{L}_1(t')$ and show that in each case r_i becomes a terminal robot. Different scenarios are as follows:

- We first consider a basic scenario in which (i) $\mathcal{L}_1(t')$ contains exactly one junction robot position $r_k(t')$ and (ii) r_k lies exactly on two lines of collinearity. Let $\mathcal{L}_2(t') (\neq \mathcal{L}_1(t'))$ be the other line of collinearity of r_k .
 - * Suppose $r_k(t')$ is the only junction robot position on $\mathcal{L}_2(t')$. Then, as in case-1, within finite number of rounds r_k becomes a terminal robot (Fig. 2). Once r_k becomes terminal, again by case-1, the collinearity among the robots initially on $\mathcal{L}_1(t')$ are broken within finite number of rounds and r_i becomes terminal.
 - * Suppose $\mathcal{L}_2(t')$ contains another junction robot r_m and r_k and r_m are the only two robots which occupy junction position on $\mathcal{L}_2(t')$. Let $\mathcal{L}_3(t') (\neq \mathcal{L}_2(t'))$ be the line of collinearity on which r_m lies. If r_m lies

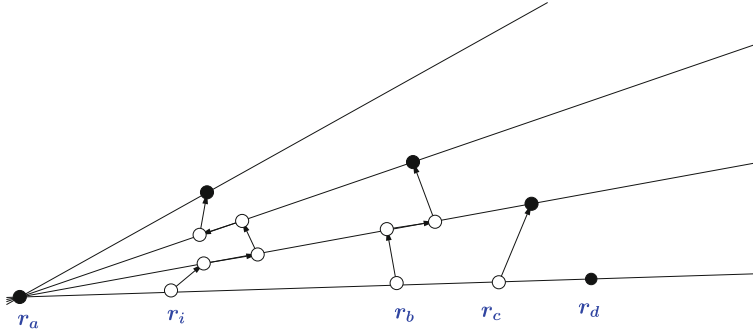


Fig. 1. An illustration of case-1 of Lemma 2: non-terminal robots on a line of collinearity $\mathcal{L}_1(t')$, containing no junction robot position, become terminal within finite number of movements: dark circles are current positions of the robots and white circles are old positions of the robots, the arrows show the directions of movements of the robots

on exactly two lines of collinearity $\mathcal{L}_2(t')$ and $\mathcal{L}_3(t')$ and $\mathcal{L}_3(t')$ does not contain any other junction robot position, by the same arguments as above, within finite number of rounds r_i becomes terminal.

* Suppose $\mathcal{L}_3(t')$ contains another junction robot position. Continuing our arguments as above, we get a sequence \mathcal{S} of lines of collinearity. Since there are finite number of robots, this sequence either ends with a line of collinearity $\mathcal{L}_k(t')$ containing exactly one junction robot position or it contains a *cycle*. If former is true, as above, all the non-terminal robots in this sequence become terminal within finite time. When \mathcal{S} contains a *cycle*, then a type-1 move breaks this *cycle* and r_i becomes terminal within finite time.

Thus, in these basic scenarios within finite number of rounds, r_i becomes terminal.

- Now consider the general scenario, in which a line of collinearity may contain more than two junction robot positions. Starting from $\mathcal{L}_1(t')$, we can get many such sequences of lines of collinearity as described above. Let $\tilde{\mathcal{S}}$ denote the set of all these sequence. Since the sequences in $\tilde{\mathcal{S}}$ may have common lines, breaking of collinearities from one line may depend on breaking of collinearities from another line.

* If no sequence in $\tilde{\mathcal{S}}$ contains a *cycle*, then type-0 movements i.e., movements not along the lines of collinearity will break all the collinearities in $\tilde{\mathcal{S}}$.

* Suppose a sequence in $\tilde{\mathcal{S}}$ contains a *cycle*, say \mathcal{C} . Let r_x be a robot at a critical robot position on a line \mathcal{L}_u in \mathcal{C} . If r_x makes a type-1 move along \mathcal{L}_u , then r_x does not remain a robot at critical position and cycle \mathcal{C} is broken. Suppose r_x makes a type-1 move along another line of collinearity \mathcal{L}_v . If \mathcal{L}_v does not belong to a *cycle*, by case-1, within finite rounds, r_x does not remain non-terminal with the robots on \mathcal{L}_v and when r_x makes a type-1 move along \mathcal{L}_u , it breaks the cycle \mathcal{C} . If \mathcal{L}_v

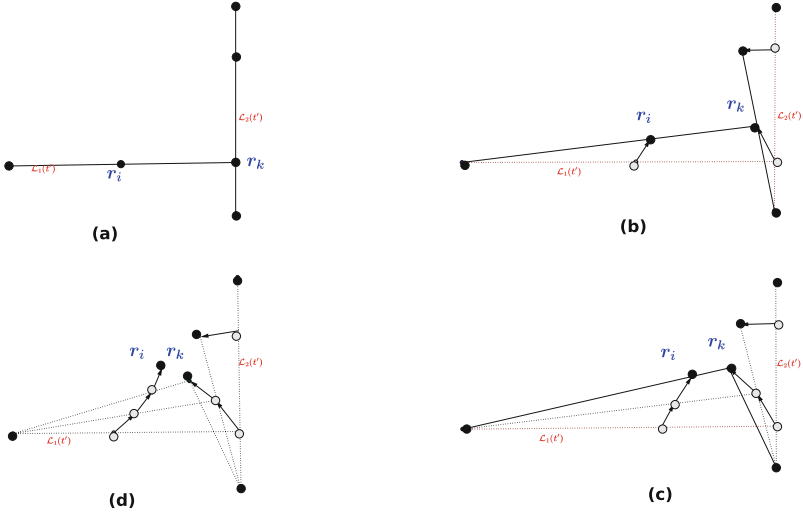


Fig. 2. An illustration of case-2 of Lemma 2: (a) $\mathcal{L}_1(t')$ and $\mathcal{L}_2(t')$ contain exactly one junction robot position $r_k(t')$, (b)-(d) demonstrate a sequence of type-0 movements for the robots to show that r_i becomes terminal within finite number of rounds: dark circles are current positions of the robots and white circles are old positions of the robots, the arrows show the directions of movements of the robots

belongs to a *cycle*, r_x is a robot at a critical robot position on \mathcal{L}_v and a type-1 movement of r_x along \mathcal{L}_v breaks this cycle. Thus, within finite time all the cycles in $\tilde{\mathcal{S}}$ is broken.

Hence, within finite time, r_i becomes a terminal robot. Since robots are semi-synchronous, by Lemma 1, r_i remains as terminal once it becomes so.

Lemma 3. *The movements of the robots are collision-free.*

Proof. Let r_i and r_j be two arbitrary robots and at least one of them moves. Consider a robot r_k visible to at least one of r_i and r_j . If r_i and r_j collide, then r_i , r_j and r_k would become collinear or remain collinear which contradicts Lemmas 1 and 2. This implies that the movements of the robots are collision free during the whole execution of *MutualVisibility()*.

Lemma 4. *If $\mathcal{R}(t_0) \notin \tilde{\mathcal{C}}_L$, during the whole execution of algorithm *MutualVisibility()*, the convex hull of the robot positions in $\mathcal{R}(t_0)$ remains invariant in size and shape.*

Proof. Let $\mathcal{CH}(t_0)$ denote the convex hull of $\mathcal{R}(t_0)$. The robots occupying the vertices of $\mathcal{CH}(t_0)$ are terminal robots. According to algorithm *MutualVisibility()*, these robots do not move. The robots on the edges of $\mathcal{CH}(t_0)$ move inside the convex hull $\mathcal{CH}(t_0)$ and no robot, lying inside the hull, crosses

any edge of the convex hull (as per definitions of directions of movements and amount of displacement in case-1 of subsection D). Hence, $\mathcal{CH}(t_0)$ remains invariant in size and shape.

Lemma 5. *Algorithm $MutualVisibility()$ is self-stabilizing.*

Proof. Robots use type-1 movements to break the *cycles*. The type-0 movements are used to break the other type of collinearities. In our approach, robots start with type-0 movements. If a robot is not a *critical* robot and starts with a type-1 movement, it would take at most one additional round to become a terminal robot. On the other hand, if a *critical* robot starts with a type-1 movement, it breaks the cycle and would take one round less to become a terminal robot. Thus, our approach works even if robots start with any value in their internal.

From the above lemmas, we have the following theorem:

Theorem 1. *Algorithm $MutualVisibility()$ provides a self-Stabilization solution to the mutual visibility problem without any collision for a set of semi-synchronous, communication-less robots, placed in distinct location, with 1 bit of persistent memory.*

4 Conclusion

This paper presents a self-stabilizing distributed algorithm to solve the mutual visibility problem in finite time for a set of communication-less semi-synchronous robots endowed with a constant amount of persistent memory. The proposed algorithm uses only 1 bit of persistent memory. The robots use their persistent memories only to remember information about their last movements. There is no explicit communication between the robots. The algorithm also guarantees collision-free movements for the robots. The proposed algorithm also maintains the convex hull of the initial robot positions. The results of this paper leave many open questions. How does the internal persistent memory can help to reduce the communication overheads in the existing solutions for the mutual visibility problem, where external lights are used for communicating the internal states of the robots? How to solve the mutual visibility problem for asynchronous robots in this setting? What would be the impact of internal persistent memory in the solutions of other geometric problems?

References

1. Das, S., Flocchini, P., Prencipe, G., Santoro, N., Yamashita, M.: The power of lights: synchronizing asynchronous robots using visible bits. In: Proceedings of 32nd International Conference on Distributed Computing Systems (ICDCS), pp. 506–515 (2012)
2. Bhagat, S., Chaudhuri, S.G., Mukhopadhyaya, K.: Formation of general position by asynchronous mobile robots under one-axis agreement. In: Kaykobad, M., Petreschi, R. (eds.) WALCOM 2016. LNCS, vol. 9627, pp. 80–91. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-30139-6_7

3. Flocchini, P., Prencipe, G., Santoro, N.: Distributed Computing by Oblivious Mobile Robots. Morgan & Claypool, San Rafael (2012)
4. Flocchini, P., Santoro, N., Viglietta, G., Yamashita, M.: Rendezvous of two robots with constant memory. In: Moscibroda, T., Rescigno, A.A. (eds.) SIROCCO 2013. LNCS, vol. 8179, pp. 189–200. Springer, Cham (2013). https://doi.org/10.1007/978-3-319-03578-9_16
5. Aljohani, A., Sharma, G.: Complete visibility for mobile robots with lights tolerating faults. *Int. J. Netw. Comput.* **8**(1), 32–52 (2018)
6. Aljohani, A., Poudel, P., Sharma, G.: Fault-tolerant complete visibility for asynchronous robots with lights under one-axis agreement. In: Rahman, M.S., Sung, W.-K., Uehara, R. (eds.) WALCOM 2018. LNCS, vol. 10755, pp. 169–182. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-75172-6_15
7. Sharma, G., Alsaedi, R., Busch, C., Mukhopadhyay, S.: The complete visibility problem for fat robots with lights. In: Proceedings of 19th International Conference on Distributed Computing and Networking (ICDCN 2018), p. 21 (2018)
8. Sharma, G., Vaidyanathan, R., Trahan, J.L.: Constant-time complete visibility for asynchronous robots with lights. In: Spirakis, P., Tsigas, P. (eds.) SSS 2017. LNCS, vol. 10616, pp. 265–281. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69084-1_18
9. Bhagat, S., Mukhopadhyaya, K.: Optimum algorithm for mutual visibility among asynchronous robots with lights. In: Spirakis, P., Tsigas, P. (eds.) SSS 2017. LNCS, vol. 10616, pp. 341–355. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69084-1_24
10. Sharma, G., Busch, C., Mukhopadhyay, S.: Mutual visibility with an optimal number of colors. In: Bose, P., Gašieniec, L.A., Römer, K., Wattenhofer, R. (eds.) ALGOSENSORS 2015. LNCS, vol. 9536, pp. 196–210. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-28472-9_15
11. Sharma, G., Busch, C., Mukhopadhyay, S.: Bounds on mutual visibility algorithms. In: Proceedings of 27th Canadian Conference on Computational Geometry (CCCG 2015) (2015)
12. Di Luna, G.A., Flocchini, P., Gan Chaudhuri, S., Poloni, F., Santoro, N., Viglietta, G.: Mutual visibility by luminous robots without collisions. In: *Information and Computation*, vol. 254, pp. 392–418 (2017)
13. Di Luna, G.A., Flocchini, P., Poloni, F., Santoro, N., Viglietta, G.: The mutual visibility problem for oblivious robots. In: Proceedings of 26th Canadian Conference on Computational Geometry (CCCG 2014) (2014)
14. Sharma, G.: Mutual visibility for robots with lights tolerating light faults. In: Proceedings of IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), pp. 829–836 (2018)



Pushing the Online Matrix-Vector Conjecture Off-Line and Identifying Its Easy Cases

Leszek Gąsieniec¹, Jesper Jansson², Christos Levcopoulos³,
Andrzej Lingas^{3(✉)}, and Mia Persson⁴

¹ Department of Computer Science, University of Liverpool,
Ashton Street, Liverpool L69 38X, UK

L.A.Gasieniec@liverpool.ac.uk

² Department of Computing, The Hong Kong Polytechnic University, Hung Hom,
Kowloon, Hong Kong

jesper.jansson@polyu.edu.hk

³ Department of Computer Science, Lund University, 22100 Lund, Sweden

{Christos.Levcopoulos, Andrzej.Lingas}@cs.lth.se

⁴ Department of Computer Science and Media Technology, Malmö University,
20506 Malmö, Sweden

mia.persson@mau.se

Abstract. Henzinger et al. posed the so called Online Boolean Matrix-vector Multiplication (OMv) conjecture and showed that it implies tight hardness results for several basic partially dynamic or dynamic problems [STOC'15].

We show that the OMv conjecture is implied by a simple off-line conjecture. If a not uniform (i.e., it might be different for different matrices) polynomial-time preprocessing of the matrix in the OMv conjecture is allowed then we can show such a variant of the OMv conjecture to be equivalent to our off-line conjecture. On the other hand, we show that the OMV conjecture does not hold in the restricted cases when the rows of the matrix or the input vectors are clustered.

1 Introduction

Henzinger et al. considered the following *Online Boolean Matrix-vector Multiplication* (OMv) problem in [8]. Initially, there are given an integer n and an $n \times n$ Boolean matrix M . Then, for $i = 1, \dots, n$, in the i -th round there is given an n -dimensional Boolean column vector v_i , and the task is to compute the product of M with v_i before the next round. The objective is to design a (possibly randomized) algorithm that solves the OMv problem, i.e., it computes all the n products as quickly as possible. In [8], Henzinger et al. provided efficient reductions of the OMv problem to several basic partially dynamic or dynamic problems including subgraph connectivity, Pagh's problem, d -failure connectivity, decremental single-source shortest paths, and decremental transitive closure.

They also stated the following OMv conjecture in [8].

Conjecture 1 OMv conjecture. For any constant $\epsilon > 0$, there is no randomized algorithm that solves the OMv problem in $O(n^{3-\epsilon})$ time with an error probability of at most $1/3$.

Their conjecture implies tight hardness results for the aforementioned partially dynamic or dynamic problems [8]. It also implies the following off-line Mv conjecture [8].

Conjecture 2 Mv conjecture. For any constant $\epsilon > 0$ and any polynomial $p(\cdot)$, there is no randomized preprocessing and randomized algorithm such that any $n \times n$ Boolean matrix M can be preprocessed in $p(n)$ time so the Boolean product of M with an arbitrary Boolean n -dimensional column vector can be computed in $O(n^{2-\epsilon})$ time with an error probability of at most $1/3$.

The fastest known algorithm for the OMv problem is due to Green Larsen and Williams [7]. Their recent (not combinatorial) randomized algorithm runs in $O(n^3/2^{\Omega(\sqrt{\log n})})$ time. Williams has also shown in [10] that any $n \times n$ Boolean matrix can be preprocessed in $O(n^{2+\epsilon})$ time so the Boolean product of the matrix with an arbitrary n -dimensional Boolean vector can be computed in $O(n^2/\log^2 n)$ time. This implies that the Mv problem corresponding to the Mv conjecture admits an $O(n^2/\log^2 n)$ -time solution. Also recently, Chakraborty et al. have established tight cell probe bounds for succinct Boolean matrix-vector multiplication in [4].

Our Contributions. We show that the OMv conjecture is implied by the following simple off-line MvP conjecture: For any constant $\epsilon > 0$ and any polynomial p there is an $n \times n$ Boolean matrix M that cannot be preprocessed in $p(n)$ time such that the Boolean product of M with an arbitrary n -dimensional column vector v can be computed in $O(n^{2-\epsilon})$ time with an error probability of at most $1/3$. There is a subtle but a substantial difference between our MvP conjecture and the Mv conjecture, the latter stated and shown to be implied by the OMv conjecture in [8]. In our conjecture the preprocessing is not uniform with respect to the matrices while in the Mv conjecture in [8] one considers a uniform, universal preprocessing. It follows that the difficulty of proving the OMv conjecture lies between the two aforementioned off-line conjectures: OMv is not more difficult than MvP and it is not easier than Mv.

We also show that if we relax the OMv problem by allowing for a not uniform polynomial-time preprocessing of the matrix M then the corresponding online conjecture will be equivalent to our MvP conjecture.

Basically, the Combinatorial Boolean Matrix Multiplication conjecture (CBMM conjecture) states that there is no combinatorial (randomized) algorithm for the Boolean product of two $n \times n$ Boolean matrices that runs in substantially subcubic time [2, 8]. Marginally, we also observe that if we strengthen the CBMM conjecture by allowing for a polynomial-time uniform preprocessing of one of the matrices, the resulting conjecture will be equivalent to the original CBMM conjecture.

On the other hand, by adapting known algorithms for Boolean matrix product of matrices with clustered data [3, 5, 6], we obtain a combinatorial randomized algorithm for the product of a Boolean $n \times n$ matrix M and an arbitrary Boolean n -dimensional column vector v running in $\tilde{O}(n + ST_M)$ time after an $\tilde{O}(n^2)$ -time preprocessing of M , where ST_M stands for the cost of a minimum spanning tree of the rows of M under the extended Hamming distance (never exceeding the Hamming distance). Consequently, we obtain a combinatorial randomized algorithm for the OMv problem running in $\tilde{O}(n(n + ST_M))$ time. We also show that OMv admits a combinatorial randomized algorithm running in $\tilde{O}(n \max\{ST(V), n^{1+o(1)}\})$ time, where $ST(V)$ stands for the cost of a minimum spanning tree of the column vectors v_1, \dots, v_n under the extended Hamming distance. The time analysis of the latter algorithm relies in part on our analysis of an approximate nearest-neighbour online heuristic for the aforementioned minimum spanning tree.

The overwhelming majority of the reductions of the OMv problem to other partially dynamic or dynamic problems in [8] are unfortunately one-way reductions that do not yield applications of our algorithms for the OMv and Mv problems. Following the applications of the Mv problem given in [2, 10], we provide analogous applications of our algorithms to vertex subset queries (e.g., for a given graph, such a query asks if a given subset of vertices is independent), triangle membership queries and 2-CNF formula evaluation queries.

Organization of the Paper. Section 2 introduces three new conjectures and it shows implications and equivalences between them and the OMv conjecture. Section 3 presents our algorithms for the OMv and Mv problems whose time complexity is expressed in terms of the minimum cost of a spanning tree of the rows of the matrix or the input column vectors under the extended Hamming distance. Section 4 presents applications of our algorithms. Section 5 concludes with some final remarks. Because of space considerations, the proof of Lemma 3 as well as a marginal subsection of Sect. 2 on Combinatorial Boolean Matrix Product, and an additional application are omitted in this extended abstract.

2 An Off-Line Conjecture

By the auxiliary Boolean Matrix-vector multiplication problem (AMv) we shall mean the problem of efficiently computing the product of a fixed $n \times n$ Boolean matrix M , that can be (not uniformly) preprocessed in $O(n^{3-\epsilon})$ time for some fixed $\epsilon > 0$, with an arbitrary n -dimensional Boolean column vector v . We state the following conjecture related to the AMv problem.

Conjecture 3 AMv conjecture. For any constant $\epsilon > 0$ and constants c_1, c_2 , there is an $n \times n$ Boolean matrix M that cannot be preprocessed in $c_1 n^{3-\epsilon}$ time such that the Boolean product of M with an arbitrary n -dimensional Boolean column vector v can be computed in $c_2 n^{2-\epsilon}$ time with an error probability of at most $1/3$.

We shall show the AMv conjecture to imply the OMv one.

Lemma 1. *Let ϵ be a positive constant and let M be an $n \times n$ Boolean matrix. If the OMv problem for M can be solved in $O(n^{3-\epsilon})$ time with an error probability of at most $1/3$ then the matrix M can be (not uniformly) preprocessed in $O(n^{3-\epsilon})$ time such that the Boolean product of M with an arbitrary input n -dimensional Boolean column vector v can be computed in $O(n^{2-\epsilon})$ time with an error probability of at most $1/3$. Consequently, the AMv conjecture implies the OMv conjecture.*

Proof. Construct a sequence of n -dimensional Boolean vectors v_1, \dots, v_n iteratively by picking as v_i a vector that jointly with the preceding vectors maximizes the total time of the assumed OMv solution for v_1, \dots, v_i . Since the assumed OMv solution for the whole sequence takes $O(n^{3-\epsilon})$ time, there must be $i \in \{1, \dots, n\}$ such that the product of M with v_i is computed in $O(n^{2-\epsilon})$ time after computing the products of M with the preceding vectors in the sequence. The computation of all the products clearly takes $O(n^{3-\epsilon})$ time and it has an error probability of at most $1/3$. By the definition of v_i , if we compute instead of the product of M with v_i , the product of M with an arbitrary n -dimensional input vector v , the computation will take only $O(n^{2-\epsilon})$ time after the products with the preceding vectors have been computed. Again, the computation of all the products, and hence in particular that of M with v , will have an error probability of at most $1/3$. Since the vectors $v_1 \dots v_{i-1}$ are fixed, the computation of the products of M with the preceding vectors can be regarded as an $O(n^{3-\epsilon})$ -time preprocessing. \square

Unfortunately, we cannot show the reverse implication, i.e., that the OMv conjecture implies the AMv one like it implies the Mv conjecture [8]. The reason is that in the definition of the AMv problem, we do not require a universal preprocessing that could work for any matrix M of size $n \times n$, we only require the existence of an individual preprocessing for a given M .

In the next lemma, we demonstrate that if we allow for an arbitrary (not uniform) polynomial-time preprocessing instead of the substantially subcubic one, we will obtain a problem equivalent to the AMv one. This lemma and its proof idea of dividing the matrix and the vector into appropriate submatrices and subvectors are similar to Lemma 2.3 and its proof idea in [8], respectively.

Lemma 2. *Let δ and ϵ be positive constants. If for any $n \times n$ Boolean matrix M there is an $O(n^{3+\delta})$ -time preprocessing such that the product of M with an arbitrary n -dimensional Boolean column vector v can be computed in $O(n^{2-\epsilon})$ time with an error probability of at most $1/3$ then there is a positive constant ϵ' such that after an $O(n^{3-\epsilon'})$ -time preprocessing the product of M with such a vector v can be computed in $O(n^{2-\epsilon'})$ time with an error probability of at most $1/3$.*

Proof. Divide M into $n^{2\alpha}$ quadratic submatrices $M_{i,j}$ of size $n^{1-\alpha} \times n^{1-\alpha}$, where $i, j \in \{1, \dots, n^\alpha\}$. Preprocess all the submatrices in $O(n^{2\alpha} \times (n^{1-\alpha})^{3+\delta})$ time. Then, the product of M with the vector v can be computed in $O(n^{2\alpha} \times (n^{1-\alpha})^{2-\epsilon} + n^{1+\alpha})$ time. The last term in the expression represents the cost of

summing the results of the products of the submatrices with respective subvectors of v of length $n^{1-\alpha}$. In order to obtain an exponent of the total preprocessing time in the form $3 - \epsilon'$ and the exponent of computing the product in the form $2 - \epsilon'$, it is sufficient to solve the inequalities $2\alpha + (1 - \alpha)(3 + \delta) < 3$, $2\alpha + (1 - \alpha)(2 - \epsilon) < 2$ and $1 + \alpha < 2$ with respect to α . Any α in the open interval $(\frac{\delta}{1+\delta}, 1)$ satisfies these inequalities.

Following the proof of Lemma 2.3 in [8], we can keep the error probability below $1/3$ by repeating the computation of each of the products of a submatrix of M with a respective vector $O(\log n)$ times, and picking the most frequent answer. In order to tackle the additional logarithmic factor in the time complexity, we can slightly decrease our ϵ' . \square

We shall call the problem and the conjecture resulting from the AMv problem and the AMv conjecture by replacing an $O(n^{3-\epsilon})$ (not uniform) preprocessing time with a polynomial (not uniform) preprocessing time, a *Boolean Matrix-vector multiplication with (polynomial-time not uniform) preprocessing problem* and a *Boolean Matrix-vector multiplication with (polynomial-time not uniform) preprocessing conjecture* (MvP for short), respectively. Since the MvP conjecture trivially implies the AMv conjecture, by Lemmata 1, 2, we obtain the following theorem.

Theorem 1. *The AMv and MvP conjectures are equivalent and they imply the OMv conjecture.*

Relaxing the OMv Problem. In order to obtain a version of OMv equivalent to AMv and MvP, we shall consider generalized versions of the OMv problem and the OMv conjecture allowing for a not uniform polynomial-time preprocessing of the matrix. We shall term them, the OMvP problem and the OMvP conjecture, respectively.

The proof of the following lemma is analogous to that of Lemma 1.

Lemma 3. *Let ϵ be a positive constant, and let M be an $n \times n$ Boolean matrix. If the OMvP problem for M and any positive natural number n , after a polynomial-time (not uniform) preprocessing of M can be solved in $O(n^{3-\epsilon})$ time with an error probability of at most $1/3$ then the matrix M can be (not uniformly) preprocessed in polynomial time such that the Boolean product of M with an arbitrary input n -dimensional Boolean column vector v can be computed in $O(n^{2-\epsilon})$ time with an error probability of at most $1/3$. Consequently, the MvP conjecture implies the OMvP conjecture.*

Lemma 4. *Let ϵ be a positive constant, and let M be an $n \times n$ Boolean matrix. If the AMv problem for M can be solved in $O(n^{2-\epsilon})$ time with an error probability of at most $1/3$ after an $O(n^{3-\epsilon})$ not uniform preprocessing of M then the OMvP problem for the matrix M and n Boolean column vectors can be solved in $O(n^{3-\epsilon})$ time with an error probability of at most $1/3$. Consequently, the OMvP conjecture implies the AMv conjecture.*

Proof. Before computing the product of M with the first vector, perform the appropriate not uniform $O(n^{3-\epsilon})$ time preprocessing of M . After that the product of M with each consecutive vector can be computed in $O(n^{2-\epsilon})$ time, so the total time for n vectors becomes $O(n^{3-\epsilon})$. We can keep the error probability below $1/3$ for the whole sequence of input vectors similarly as in the proof of Lemma 2. \square

Since the conjectures AMv and MvP are equivalent (see Theorem 1), by Lemmata 3 and 4, we obtain the following extension of Theorem 1.

Theorem 2. *The MvP, AMv and OMvP conjectures are equivalent.*

3 Easy Cases of Matrices and Vectors for the Conjectures

Björklund et al. [3] proposed a method of multiplying two Boolean matrices by using a close approximation of the minimum spanning tree of the rows or columns of one of the matrices under the Hamming distance. Subsequently, the method has been generalized to include the so called extended Hamming distance [6] and integer matrix multiplication [5]. In the first warming-up subsection, we present an explicit adaptation of the aforementioned generalizations to the case of the product of an $n \times n$ Boolean (or 0 – 1) matrix M and an n -dimensional Boolean (or 0 – 1) column vector v in the context of the OMv conjecture. Several results presented in the first subsection can be regarded as implicit in [5,6]. This is not the case in the second subsection handling the online scenario where the input column vectors are clustered. Here, we have to develop a novel online approach involving among other things an analysis of an approximate nearest-neighbour online heuristic for minimum spanning tree of the vectors under the extended Hamming distance. We shall use the following concepts in both subsections.

Definition 1. *For two 0 – 1 strings $s = s_1s_2\dots s_m$ and $u = u_1u_2\dots u_m$, their Hamming distance, i.e., the number of $k \in \{1, \dots, m\}$, s.t., $s_k \neq u_k$, is denoted by $H(s, u)$. An extended Hamming distance, $EH(s, u)$, between the strings, is defined by a recursive equation $EH(s, u) = EH(s_{l+1}\dots s_m, u_{l+1}\dots u_m) + (s_l + u_l \bmod 2)$, where l is the maximum number, s.t., $s_j = s_1$ and $u_j = u_1$ for $j = 1, \dots, l$.*

A differentiating block for the strings s, u is a maximal consecutive subsequence w of $1, 2, \dots, n$, s.t., either for each $i \in w$ $s_i = 1$ and $u_i = 0$ or for each $i \in w$ $s_i = 0$ and $u_i = 1$. In the first case, we set $h(s) = -1$ while in the second case $h(s) = 1$.

3.1 Small Spanning Tree of the Rows of the Matrix (Warming Up)

For $c \geq 1$ and a finite set S of points in a metric space, a c -approximate minimum spanning tree for S is a spanning tree in the complete weighted graph on S , with edge weights equal to the distances between the endpoints, whose total weight is at most c times the minimum.

Fact 1 (Lemma 3 in [6]). For $\epsilon > 0$, a $2 + \epsilon$ -approximate minimum spanning tree for a set of n $0 - 1$ strings of length d under the extended Hamming metric can be computed by a Monte Carlo algorithm in time $O(dn^{1+1/(1+\epsilon/2)})$.

By selecting $\epsilon = 2 \log n$, we obtain the following lemma.

Lemma 5. Let M be an $n \times n$ Boolean matrix. An $O(\log n)$ -approximation minimum spanning tree for the set of rows of M under the extended Hamming distance can be constructed by a Monte Carlo algorithm in $\tilde{O}(n^2)$ time.

We shall also use the following data structure, easily obtained by computing all prefix sums:

Fact 2 (e.g., see [5]). For a sequence of integers a_1, a_2, \dots, a_n , one can construct a data structure that supports a query asking for reporting the sum $\sum_{k=i}^j a_k$ for $1 \leq i \leq j \leq n$ in $O(1)$ time. The construction takes $O(n)$ time.

By using Lemma 5 and Fact 2, we obtain the following algorithm which in fact computes the arithmetic product of the input Boolean matrix M and Boolean vector v interpreted as $0 - 1$ ones. Observe that the aforementioned arithmetic product immediately yields the corresponding Boolean one.

Algorithm 1

Input: An $n \times n$ Boolean matrix M and an n -dimensional Boolean column vector v .

Output: The arithmetic product $c = (c_1, \dots, c_n)$ of M and v interpreted as a $0 - 1$ matrix and a $0 - 1$ vector, respectively.

1. Find an $O(\log n)$ -approximate spanning tree T for the rows $row_i(M)$, $i = 1, \dots, n$, of M under the extended Hamming distance and a traversal (i.e., a not necessarily simple path visiting all vertices) of T .
2. For any pair $(row_i(M), row_l(M))$, where the latter row follows the former in the traversal, find the differentiating blocks s for $row_i(M)$ and $row_l(M)$ and as well as the differences $h(s)$ (1 or -1) between the common value of each entry in $M_{l, \min s}, \dots, M_{l, \max s}$ and the common value of each entry in $M_{i, \min s}, \dots, M_{i, \max s}$.
3. Initialize a data structure D for counting partial sums of the values of coordinates on continuous fragments of the vector v .
4. Iterate the following steps:
 - (a) Compute c_q where q is the index of the row from which the traversal of T starts.
 - (b) While following the traversal of T , iterate the following steps:
 - i. Set i, l to the indices of the previously traversed row and the currently traversed row, respectively.
 - ii. Set c_l to c_i .
 - iii. For each differentiating block s for $row_i(M)$ and $row_l(M)$, compute $\sum_{k \in s} v_k$ using D and set c_l to $c_l + h(s) \sum_{k \in s} v_k$.
5. Output the vector (c_1, c_2, \dots, c_n) .

Definition 2. For an $n \times n$ Boolean matrix A , let ST_A stand for the minimum cost of a spanning tree of $row_i(A)$, $i \in \{1, \dots, n\}$, under the extended Hamming distance.

Lemma 6. Algorithm 1 runs in $\tilde{O}(n^2 + ST_M)$ with high probability. If Steps 1, 2, 3 are treated as a preprocessing of the matrix M then it runs in $\tilde{O}(n + ST_M)$ time after an $\tilde{O}(n^2)$ -time preprocessing.

Proof. The approximate minimum spanning tree T in Step 1 can be constructed by a Monte Carlo algorithm in $\tilde{O}(n^2)$ time by Lemma 5. Its traversal can be easily found in $O(n)$ time. Since the length of the traversal is linear in n , Step 2 can be easily implemented in $O(n^2)$ time. Step 3 takes $O(n)$ time by Fact 2. Finally, based on Step 2, Step 4 (b)-ii takes $\tilde{O}(1 + EH(row_i(M), row_l(M)))$ time. Let U stand for the set of directed edges forming the traversal of the spanning tree T . It follows that Step 4 (b) can be implemented in $\tilde{O}(n + \sum_{(i,l) \in U} EH(row_i(M), row_l(M)))$ time, i.e., in $\tilde{O}(n + ST_M)$ time by Lemma 5. Consequently, Step 4 takes $\tilde{O}(n + ST_M)$ time. \square

By Lemma 6, we obtain:

Theorem 3. The Boolean product c of an $n \times n$ Boolean matrix M and an n -dimensional Boolean column vector v can be computed in $\tilde{O}(n + ST_M)$ time with high probability after $\tilde{O}(n^2)$ -time preprocessing.

Proof. The correctness of Algorithm 1 follows from the observation that a differentiating block s for $row_i(M)$ and $row_l(M)$ yields the difference $h(s) \sum_{k \in s} v_k$ between c_l and c_i just on the fragment corresponding to $M_{i, \min s}, \dots, M_{i, \max s}$ and $M_{l, \min s}, \dots, M_{l, \max}$, respectively. Lemma 6 yields the upper bounds in terms of ST_M . \square

Corollary 1. The OMv problem for an $n \times n$ Boolean matrix can be solved in $\tilde{O}(n(n + ST_M))$ time while the Mv problem can be solved in $\tilde{O}(n + ST_M)$ time after $\tilde{O}(n^2)$ -time preprocessing.

3.2 Small Spanning Tree of the Input Vectors

In this subsection, we assume an online scenario where besides the Boolean matrix there is given a sequence of n -dimensional Boolean vectors received one at a time. In order to specify and analyze our algorithm, we need the following concepts and facts on them.

Definition 3. For a metric space P and a point $q \in P$, an c -approximate nearest neighbour of q in P is a point $p \in P$ different from q such that for all $p' \in P, p' \neq q$, $dist(p, q) \leq c \times dist(p', q)$. The ϵ -approximate nearest neighbour search problem (ϵ -NNS) in P is to find for a query point $q \in P$ a $(1 + \epsilon)$ -approximate nearest neighbour of q in P .

Fact 3 (See 3rd row in Table 4.3.1.1 in [1]). For $\epsilon > 0$, there is a Monte Carlo algorithm for the dynamic ϵ -NNS in $\{0, 1\}^d$ under the Hamming metric which requires $O(d\ell^{\frac{1}{1+2\epsilon} + o(1)})$ query time and $O(d\ell^{\frac{1}{1+2\epsilon} + o(1)})$ update time, where ℓ is the maximum number of stored vectors in $\{0, 1\}^d$.

Fact 4 [6]. There is a simple, linear-time, transformation of any 0 – 1 string w into the string $t(w)$ such that for any two 0 – 1 strings s and u , $EH(s, u) = \lceil \frac{H(t(s), t(u))}{2} \rceil$.

By combining Facts 3, 4, we obtain the following corollary.

Corollary 2. There is a randomized Monte Carlo algorithm for a dynamic $O(\log \ell)$ -NNS in $\{0, 1\}^d$ under the extended Hamming metric which requires $O(d\ell^{o(1)})$ query time and $O(d\ell^{o(1)})$ update time.

Our online algorithm is as follows.

Algorithm 2

Input: Given a priori an $n \times n$ Boolean matrix M and an online sequence of n -dimensional Boolean vectors v_1, v_2, \dots, v_ℓ received one at a time.

Output: For $i = 1, \dots, \ell$, the arithmetic product $c^i = (c_1^i, \dots, c_n^i) = Mv_i$ of M and v_i , treated as a 0-1 matrix and a 0-1 column vector, is output before receiving v_{i+1} .

1. For $j = 2, \dots, n$, initialize a data structure D_j that for any interval $s \subseteq \{1, \dots, n\}$ reports $\sum_{k \in s} M[j, k]$ using Fact 2.
2. Receive the first vector v_1 and compute the arithmetic product $c^1 = (c_1^1, \dots, c_n^1)$ of M with v_1 by the definition.
3. For $i = 2, \dots, \ell$, receive the i -th vector $v_i = (v_i^1, \dots, v_i^n)$ and iterate the following steps:
 - (a) Find an $O(\log \ell)$ -approximate nearest neighbour v_m of v_i in the set $\{v_1, \dots, v_{i-1}\}$.
 - (b) Determine the differentiating blocks s and the differences $h(s)$ for v_m and v_i .
 - (c) For $j = 1, \dots, n$ iterate the following steps.
 - i. Set c_j^i to c_j^m .
 - ii. For each differentiating block s of v_m and v_i iterate the following steps.
 - A. Compute $\sum_{k \in s} M[j, k]$ using D_j .
 - B. Set c_j^i to $c_j^i + h(s) \sum_{k \in s} M[j, k]$.
 - (d) Output $c^i = (c_1^i, \dots, c_n^i)$

In the following lemmata, we analyze the time complexity of Algorithm 2. The first lemma is an immediate consequence of Corollary 2.

Lemma 7. *There is a randomized Monte Carlo algorithm for a dynamic $O(\log \ell)$ -NNS in $\{0, 1\}^d$ under the extended Hamming metric such that:*

- *The insertions of the vectors v_1 through v_ℓ in Algorithm 2 can be implemented in $O(n\ell^{1+o(1)})$ total time.*
- *The $O(\log \ell)$ -approximate nearest neighbours of v_i , $i = 2, \dots, \ell$, in $\{v_1, \dots, v_{i-1}\}$, in Step 3 (a) of Algorithm 2 can be found with high probability in $O(n\ell^{1+o(1)})$ total time.*

Proof. By Corollary 2, the $\ell - 1$ updates and $\ell - 2$ $O(\log \ell)$ -approximate nearest neighbour queries take $O(n\ell^{1+o(1)})$ total time. □

Lemma 8. *The preprocessing of the matrix M in Step 1 and computing the arithmetic product of M with v_1 in Step 2 takes $O(n^2)$ time. After that, Algorithm 2 for $i = 2, \dots, \ell$, computes the arithmetic product c^i of M and v_i before receiving v_{i+1} in $\tilde{O}(n(1 + \min\{\text{dist}(v_i, v_j) | j < i\})) + t(i)$ time, where $t(i)$ is the time taken by finding an $O(\log \ell)$ -approximate neighbour of v_i in $\{v_1, v_2, \dots, v_{i-1}\}$ and inserting v_i in the supporting data structure, with high probability. The total time is $\tilde{O}(n(\ell + ST(V))) + \sum_{i=2}^{\ell} t(i)$, where $ST(V)$ is the minimum cost of the spanning tree of the vectors in $V = \{v_1, v_2, \dots, v_\ell\}$ under the extended Hamming distance.*

Proof. Step 1 can be implemented in $O(n^2)$ time by Fact 2 while Step 2 can be easily done in $O(n^2)$ time by the definition. Step 3 (a) takes $t(i)$ time by our assumptions. The differentiating blocks s and the differences $h(s)$ for v_m and v_i can be easily determined in $O(n)$ time in Step 3 (b). Finally, since the number of the aforementioned blocks is within a polylogarithmic factor of $\min\{\text{dist}(v_i, v_j) | j < i\}$, the whole update of c^m to c^i in Step 3 (c) takes $\tilde{O}(n(1 + \min\{\text{dist}(v_i, v_j) | j < i\}))$ time. □

In order to pursue our time analysis of Algorithm 2, we need to specify and analyze the following simple online approximation heuristic for minimum spanning tree (MST).

Approximate Nearest-Neighbour Heuristic for MST

Input: an online sequence V of vectors v_1, v_2, \dots received one at time.

Output: a sequence of spanning trees T_i ' of the vectors v_1 through v_i constructed before receiving v_{i+1} for all i .

for each new vector v_i **do**

find an $f(i)$ -approximate nearest neighbour u of v_i in the set of vectors received so far;

expand the spanning tree built for the vectors received before v by $\{u, v_i\}$.

Theorem 4. *Assume that the function f is not decreasing and the input vectors to the approximate nearest-neighbour heuristic for MST are drawn from a metric space. The spanning tree constructed by the heuristic for the first t vectors has cost not exceeding $\lceil \log_2 t \rceil f(t)$ times the minimum.*

Proof. Assume first that t is a power of two. Let $V = \{v_1, \dots, v_t\}$ be the sequence of t vectors received, where v_i is the i -th vector received.

Consider a minimum cost perfect matching P of V . For each edge $\{v_i, v_j\}$ in P , where $i < j$, the cost of connecting v_j to the current spanning tree T_{i-1} of v_1 through v_{j-1} does not exceed $f(t) \times \text{dist}(v_i, v_j)$. Thus, for $t/2$ vectors v_i in V , the cost of connecting them to the current spanning tree T_{t-1} does not exceed the total cost of P times $f(t)$. It is well known that the total cost of P is not greater than half the minimum cost $TSP(V)$ of the travelling salesperson tour of V .

In order to estimate from above the cost of connecting the remaining $t/2$ vectors to the current spanning trees, we iterate our argument.

Thus, let V_1 denote the remaining set of vertices and let P_1 be their minimum-cost perfect matching. We can again estimate the cost of connecting half of the $t/2$ vectors in V_1 to the current spanning trees by the cost of P_1 times $f(t)$. On the other hand, we can estimate the cost of P_1 by $\frac{1}{2}TSP(V_1) \leq \frac{1}{2}TSP(V)$. We handle analogously the remaining $t/4$ vectors and so on. After $\log_2 t$ iterations, we are left with the first vector, and can estimate the total cost of connecting all other vectors to the current spanning trees by $\log_2 t f(t)TSP(V)/2$. On the other hand, by the doubling MST heuristic, we know that $TSP(V)$ is at most twice the cost $ST(V)$ of minimum-cost spanning tree of V . We conclude that the cost of the spanning tree of V constructed by the approximate nearest-neighbour heuristic does not exceed $\log_2 t f(t)ST(V)$.

If t is not a power of two, we have to consider minimum-cost maximum cardinality matchings instead of minimum-cost perfect matchings. Let $t' = 2^{\lceil \log_2 t \rceil}$. Observe that the number of the remaining vectors after each iteration when we start with a sequence of t vectors will be not greater than that when we start with a sequence of t' vectors having the sequence of t vectors as a prefix. This completes the proof of the $\lceil \log_2 t \rceil f(t)ST(V)$ upper bound. \square

In the special case when $f(\cdot) \equiv 1$, our online heuristic for MST in a way coincides with the greedy one for incremental minimum Steiner tree from [9], which on weighted graphs satisfying triangle inequality could be easily adapted to consider only received vertices. Hence, in this case a logarithmic upper bound on approximation factor could be also deduced from Theorem 3.2 in [9]. Putting together our lemmata and theorem in this subsection, we obtain our main result here.

Theorem 5. *Let M be an $n \times n$ Boolean matrix. For an online sequence V of n -dimensional Boolean vectors v_1, v_2, \dots, v_ℓ received one at time, the Boolean products Mv_i of M and v_i can be computed before receiving v_{i+1} . in total time $\tilde{O}(n(\ell^{1+o(1)} + ST(V)))$ with high probability by a randomized algorithm, where $ST(V)$ is the minimum cost of the spanning tree of the vectors in V under the extended Hamming distance.*

Proof. The correctness of Algorithm 2 follows from the observation that a differentiating block s for v_m and v_i yields the difference $h(s) \sum_{k \in s} M[j, k]$ between c_j^m and c_j^i just on the fragments $v_{\min s}^m, \dots, v_{i, \max s}^m$ and $v_{\min s}^i, \dots, v_{\max s}^i$, respectively. Lemmata 7, 8 and Theorem 4 yield the upper bounds in terms of $ST(V)$. \square

4 Applications to Graph Queries

Suppose that we are given a graph $G = (V, E)$ on n vertices and a subset S of V . In [10] Williams observed that the questions if S is a dominating set, an independent set, or a vertex cover in G , can be easily answered by computing the Boolean product of the adjacency matrix of G with appropriate Boolean vectors. Hence, he could conclude (Corollary 3.1 in [10]) that these questions can be answered in $O(n^2/(\epsilon \log n)^2)$ time after an $O(n^{2+\epsilon})$ preprocessing of G by using his method of multiplying $n \times n$ Boolean matrix with an n -dimensional column vector in $O(n^2/(\epsilon \log n)^2)$ time after an $O(n^{2+\epsilon})$ -time preprocessing of the matrix. By plugging in our method of Boolean matrix-vector multiplication (Theorem 3) instead, we obtain the following result.

Corollary 3. *A graph G on n vertices can be preprocessed in $\tilde{O}(n^2)$ time such that one can determine if a given subset of vertices in G is a dominating set, an independent set, or a vertex cover of G in $\tilde{O}(n + ST_G)$ time with high probability, where ST_G is the minimum cost of a spanning tree of the rows of the adjacency matrix of G under the extended Hamming distance. Using the same preprocessing, one can determine if a query vertex belongs to a triangle in G in $\tilde{O}(n + ST_G)$ time with high probability.*

To obtain corresponding applications of the results from Subsect. 3.2, we need to consider the online versions of the graph subset queries. Thus, we are given a graph G on n vertices and an online sequence of subsets S_1, \dots, S_ℓ of vertices in G . The task is to preprocess G first and then to determine for $i = 1, \dots, \ell$, if S_i is a dominating set, an independent set, or a vertex cover of G , respectively, before S_{i+1} has been received.

Corollary 4. *A graph G on n vertices can be preprocessed in $O(n^2)$ time such that for an online sequence S of subsets S_1, \dots, S_ℓ of vertices in G , for $i = 1, \dots, \ell$, one can determine if S_i is a dominating set, an independent set, or a vertex cover of G before receiving S_{i+1} (in $i < \ell$ case) in $\tilde{O}(n(\ell^{1+o(1)} + ST_S))$ total time with high probability, where ST_S is the minimum cost of a spanning tree of the characteristic vectors representing the subsets in S under the extended Hamming distance. Using the same preprocessing, for an online sequence v_1, \dots, v_ℓ of query vertices, for $i = 1, \dots, \ell$, one can determine if v_i belongs to a triangle in G before receiving v_{i+1} (in case $i < \ell$) in $\tilde{O}(n(\ell^{1+o(1)} + ST_G))$ total time with high probability.*

Proof. Recall that a subset S_i of vertices in G can be easily represented by an n -dimensional Boolean column vector w_i with 1 on the j -th coordinate iff the j -th vertex belongs to S_i . Then, S_i is independent in G iff the vector u_i resulting from multiplying the adjacency matrix of G with w_i has zeros on the coordinates corresponding to the vertices in S_i . Next, S_i is a dominating set of G iff each vertex in $V \setminus S_i$ has a neighbour in S_i , i.e., iff u_i has ones on the coordinates corresponding to vertices in $V \setminus S_i$. Finally, S_i is a vertex cover of G iff $V \setminus S_i$ is an independent set of G , i.e., iff the vector resulting from multiplying the

adjacency matrix of G with the complement of w_i has zeros on the coordinates corresponding to the vertices in $V \setminus S_i$.

Hence, it is sufficient to plug in our solution given in Theorem 5 to obtain the theorem. The preprocessing of G consists just in the construction of its adjacency matrix in $O(n^2)$ time. Note also that the extended Hamming distance between two 0–1 strings is equal to the extended Hamming distance between between the complements of these two strings. Thus, the upper bound in terms of ST_S is also valid in case of vertex cover. \square

5 Final Remarks

Our results in Sect. 3 imply that to prove the conjectures OMv, AMv and MvP it is sufficient to consider $n \times n$ Boolean matrices where ST_M is almost quadratic in n .

Interestingly enough, our approximate nearest-neighbour heuristic for MST combined with the standard MST doubling and shortcuttings techniques immediately yields a corresponding online heuristic for TSP in metric spaces. By Theorem 4, it provides TSP tours TSP_s of length at most $2^{\lceil \log_2 s \rceil} f(s)$ times larger than the optimum, where s is the number of input vectors and $f(s)$ is an upper bound on the approximation factor in the approximate nearest neighbour subroutine. The resulting TSP heuristic for $i = 2, \dots$ simply finds an $f(i)$ -nearest neighbour u of the new vector v_i and replaces the edge between u and its predecessor w by the path $\{w, v_i\}$, $\{v_i, u\}$ in TSP_{i-1} in order to obtain TSP_i .

Acknowledgements. CL, JJ and MP were supported in part by Swedish Research Council grant 621-2017-03750.

References

1. Andoni, A., Indyk, P.: Nearest neighbours in high-dimensional spaces. In: Goodman, J.E., O’Rourke, J., Toth, C.D. (eds.) Handbook of Discrete and Computational Geometry, 3rd edn. CRC Press, Boca Raton (2017)
2. Bansal, N., Williams, R.: Regularity lemmas and combinatorial algorithms. *Theory Comput.* **8**(1), 69–94 (2012)
3. Björklund, A., Lingas, A.: Fast Boolean matrix multiplication for highly clustered data. In: Dehne, F., Sack, J.-R., Tamassia, R. (eds.) WADS 2001. LNCS, vol. 2125, pp. 258–263. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44634-6_24
4. Chakraborty, D., Kamma, L., Larsen, K.G.: Tight cell probe bounds for succinct Boolean matrix-vector multiplication. In: Proceedings of STOC 2018, pp. 1297–1306 (2018)
5. Floderus, P., Jansson, J., Levkopoulos, C., Lingas, A., Sledneu, D.: 3D rectangulations and geometric matrix multiplication. *Algorithmica* **80**(1), 136–154 (2018)
6. Gašieniec, L., Lingas, A.: An improved bound on Boolean matrix multiplication for highly clustered data. In: Dehne, F., Sack, J.-R., Smid, M. (eds.) WADS 2003. LNCS, vol. 2748, pp. 329–339. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45078-8_29

7. Larsen, K.G., Williams, R.R.: Faster online matrix-vector multiplication. In: Proceedings of SODA 2017, pp. 2182–2189 (2017)
8. Henzinger, M., Krinninger, S., Nanongkai, D., Saranurak, T.: Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In: Proceedings of STOC 2015 (also presented at HALG 2016)
9. Imase, M., Waxman, B.M.: Dynamic Steiner tree problem. *SIAM J. Discrete Math.* **4**(3), 369–384 (1991)
10. Williams, R.: Matrix-vector multiplication in sub-quadratic time (some preprocessing required). In: Proceedings of SODA 2007, pp. 995–2001 (2007)



An Improved Approximation Algorithm for the k -Means Problem with Penalties

Qilong Feng, Zhen Zhang, Feng Shi, and Jianxin Wang^(✉)

School of Computer Science and Engineering, Central South University,
Changsha 410083, People's Republic of China
{csufeng, jxwang}@mail.csu.edu.cn

Abstract. The clustering problem has been paid lots of attention in various fields of compute science. However, in many applications, the existence of noisy data poses a big challenge for the clustering problem. As one way to deal with clustering problem with noisy data, clustering with penalties has been studied extensively, such as the k -median problem with penalties and the facility location problem with penalties. As far as we know, there is only one approximation algorithm for the k -means problem with penalties with ratio $25 + \epsilon$. All the previous related results for the clustering with penalties problems were based on the techniques of local search, LP-rounding, or primal-dual, which cannot be applied directly to the k -means problem with penalties to get better approximation ratio than $25 + \epsilon$. In this paper, we apply primal-dual technique to solve the k -means problem with penalties by a different rounding method, *i.e.*, employing a deterministic rounding algorithm, instead of using the randomized rounding algorithm used in the previous approximation schemes. Based on the above method, an approximation algorithm with ratio $19.849 + \epsilon$ is presented for the k -means problem with penalties.

Keywords: Approximation algorithm · k -means clustering · Primal-dual

1 Introduction

Clustering is a fundamental problem in computer science and has applications in many fields. As one of the clustering problem, k -means has been studied extensively from approximation algorithms point of view [1, 3, 7, 10, 17–19]. For a given set \mathcal{D} of n points in \mathbb{R}^d , the k -means problem aims to find a set \mathcal{S} of k cluster centers in \mathbb{R}^d such that the objective function $\sum_{j \in \mathcal{D}} \min_{i \in \mathcal{S}} c(j, i)$ is minimized, where $c(j, i)$ denotes the squared Euclidean distance between j and

This work is supported by the National Natural Science Foundation of China under Grants (61672536, 61420106009, 61872450, 61828205), Hunan Provincial Science and Technology Program (2018WK4001).

i. This problem is known to be NP-hard even for $k = 2$ [2], and even in planar space [23].

The clustering problem has an implicit assumption that all input points can be naturally clustered into several distinct groups, which may not always hold in real-world applications. Data from such applications are often contaminated with various types of noises. Charikar *et al.* [5] introduced two clustering problems dealing with noisy data: the clustering problem with outliers, and the clustering problem with penalties. For the clustering problem with outliers, lots of attention has been focused on the k -means with outliers problem [9, 13, 26], the k -median with outliers problem [6, 9, 26], the facility location with outliers problem [9], distributed clustering with outliers [11, 20], and coresets for clustering with outliers [8, 15].

For the clustering with penalties problem, each point is associated with a penalty cost. The objective of the clustering with penalties problem is to find a set \mathcal{S} of clustering centers for a given set \mathcal{D} of points in \mathbb{R}^d , such that for each point j in \mathcal{D} , j is either assigned to a center of \mathcal{S} with minimum distance to j , or discarded and the penalty cost is paid. In the past two decades, considerable efforts have been paid on the problems of clustering with penalties. For the facility location problem with penalties, Charikar *et al.* [5] gave a $(3 + \epsilon)$ -approximation by using a primal-dual approach. Jain *et al.* [16] later achieved a $(2 + \epsilon)$ -approximation based on a dual fitting technique. Xu and Xu [29] gave a $(1.853 + \epsilon)$ -approximation algorithm by integrating the techniques of primal-dual and local search. The approximation ratio was recently improved to $1.5148 + \epsilon$ via an LP-rounding algorithm [22]. Based on the assumption that the penalty costs are all equal, Wu *et al.* [27] gave a $(2.732 + \epsilon)$ -approximation for the k -median problem with penalties. Under no such assumption, Charikar *et al.* [5] gave a $(4 + \epsilon)$ -approximation by using the Lagrangian relaxation technique. Hajiaghayi *et al.* [14] later showed that the local search algorithm yields a $(3 + \epsilon)$ -approximation for the problem.

In this paper, we study the k -means problem with penalties (k -MPWP).

Definition 1 (*k -means problem with penalties* [30]). *Given a set $\mathcal{D} \subset \mathbb{R}^d$ of n points, a penalty function p defined over the points of \mathcal{D} , where $p(j) \geq 0$ for each $j \in \mathcal{D}$, and an integer $k > 0$, the task is to find a set $\mathcal{S} \subset \mathbb{R}^d$ of k centers that minimizes the objective function $\sum_{j \in \mathcal{D}} \min\{c(j, \mathcal{S}), p(j)\}$, where $c(j, \mathcal{S})$ denotes the squared distance from j to its nearest point in \mathcal{S} .*

Zhang *et al.* [30] showed that the local search approach leads to constant factor approximations for the k -MPWP. Specifically, they showed that the single-swap local search algorithm induces a $(81 + \epsilon)$ -approximation, and the algorithm that swaps up to $O(\frac{1}{\epsilon})$ centers induces a $(25 + \epsilon)$ -approximation, which is the current best ratio for the k -MPWP.

All the previous related results for the clustering with penalties problems were based on the techniques of local search, LP-rounding, or primal-dual. We briefly remark on the known techniques to illustrate the challenges in getting better ratio for the k -MPWP.

Remark 1. Local search has been applied to the k -MPWP in [30] to get the ratio $25 + \epsilon$. However, it is unclear whether this result can be improved based on more refined analysis of local search.

Remark 2. Xu and Xu [28] and Li *et al.* [22] applied LP-rounding to get approximation algorithms for the facility location problem with penalties. To the best of our knowledge, no related results based on LP-rounding for k -median with penalties are known. On the other hand, it is known that the current LP-rounding techniques yield quite large approximation ratio for the related problems of k -means. Thus, it seems hard to use LP-rounding to improve the approximation guarantee for the k -MPWP.

Remark 3. The techniques of primal-dual and Lagrangian relaxation outlined in [17] have been widely applied for the related problems of k -median [4, 6, 12, 16, 21]. In particular, Charikar *et al.* [5] gave a $(4 + \epsilon)$ -approximation for the k -median problem with penalties based on the framework of Lagrangian relaxation given in [17]. However, it can be seen that the same algorithm yields a much larger approximation ratio for k -means and its related problems. The reason is that it relies heavily on triangle inequality, which no longer holds in the squared Euclidean setting.

In this paper, primal-dual technique is applied to solve the k -MPWP. Instead of using the randomized rounding algorithm, we use a deterministic rounding algorithm that exploits the properties of the squared Euclidean metric, which is the major step to get the $19.849 + \epsilon$ ratio.

1.1 Our Results

Theorem 1. *Given an instance of the k -MPWP and a parameter $\epsilon > 0$, there is a $(19.849 + \epsilon)$ -approximation algorithm for the problem.*

We now give the general idea of our approach to solve the k -MPWP. Given an instance of the k -MPWP, we first convert it into an instance of a new problem, called the discrete k -means problem with penalties, which is to select the k centers from a discrete set $\mathcal{F} \subset \mathbb{R}^d$ of polynomial size, instead of k arbitrary points in \mathbb{R}^d . We can prove that this conversion can be implemented in polynomial time, and induces an arbitrarily small loss in the approximation ratio.

We consider the discrete k -means problem with penalties based on the Lagrangian relaxation technique given in [17]. We compute two solutions \mathcal{T}_1 and \mathcal{T}_2 for the squared Euclidean facility location problem with penalties, which is a Lagrangian relaxation of the discrete k -means problem with penalties. Assume that \mathcal{T}_1 has k_1 centers and \mathcal{T}_2 has k_2 centers, where $k_1 < k < k_2$. We are able to show that there exists a convex combination of the two solutions $a\mathcal{T}_1 + (1 - a)\mathcal{T}_2$ ($0 \leq a \leq 1$), denoted by \mathcal{T}' , such that \mathcal{T}' has exactly k centers, and has quite low cost.

Based on \mathcal{T}_1 and \mathcal{T}_2 , we get the ratio by mainly discussing how large is a , and the difference between k and k_1 .

- (1) the value of a is close to 1. We show that \mathcal{T}_1 can be used as the final solution and gives a desirable approximation.
- (2) the value of a is relative small. We prefer to choosing centers from \mathcal{T}_2 to put into the solution, since the cost of \mathcal{T}_1 might be quite expensive. The choice is made depending on the difference between k and k_1 . If the difference is upper-bounded by a small value, we show that a subset of \mathcal{T}_2 with size k can be viewed as a solution, which can be found in polynomial time. Otherwise we combine \mathcal{T}_1 and \mathcal{T}_2 into the solution based on a greedy strategy.

2 Preliminaries

Given two points j and i in \mathbb{R}^d , let $d(j, i)$ and $c(j, i)$ denote the distance and the squared distance between j and i respectively. Given a set $\mathcal{A} \subset \mathbb{R}^d$ and a point $j \in \mathbb{R}^d$, let $c(j, \mathcal{A}) = \min_{i \in \mathcal{A}} c(j, i)$ denote the squared distance from j to its nearest point in \mathcal{A} . Let $\Phi(\mathcal{A}) = \frac{1}{|\mathcal{A}|} \sum_{i \in \mathcal{A}} i$ denote the centroid of \mathcal{A} . It is known that $\Phi(\mathcal{A})$ is the optimal 1-mean solution for \mathcal{A} [18]. The minimum k -means clustering cost on \mathcal{A} is denoted by $\Delta_k(\mathcal{A})$. For a point p in \mathbb{R}^d , if p is put into the solution of the k -MPWP, then it is called that point p is opened.

We will use the following two well-known properties of the squared Euclidean metric.

Lemma 1 (weak triangle inequality). *For any three points x, y, z in \mathbb{R}^d , we have $c(x, y) \leq 2c(x, z) + 2c(z, y)$.*

Lemma 2 ([18]). *Given a set $\mathcal{A} \subset \mathbb{R}^d$ and a point $i \in \mathbb{R}^d$, we have $\sum_{j \in \mathcal{A}} c(j, i) = \sum_{j \in \mathcal{A}} c(j, \Phi(\mathcal{A})) + |\mathcal{A}|c(\Phi(\mathcal{A}), i)$.*

Matoušek *et al.* [25] gave the following concept of approximate centroid set.

Definition 2 (ϵ -approximate centroid set [25]). *Given a set $\mathcal{D} \subset \mathbb{R}^d$ and a parameter $\epsilon > 0$, a set $\mathcal{F} \subset \mathbb{R}^d$ is an ϵ -approximate centroid set for \mathcal{D} if for any set $\mathcal{A} \subseteq \mathcal{D}$, we have $\min_{i \in \mathcal{F}} \sum_{j \in \mathcal{A}} c(j, i) \leq (1 + \epsilon) \min_{i \in \mathbb{R}^d} \sum_{j \in \mathcal{A}} c(j, i)$.*

It is known that an ϵ -approximate centroid set of polynomial size can be obtained in polynomial time, for any $\epsilon > 0$ [24]. This motivates us to consider the following discrete k -means problem with penalties (discrete k -MPWP).

Definition 3 (discrete k -means problem with penalties). *Given a set $\mathcal{D} \subset \mathbb{R}^d$ of n points, a set $\mathcal{F} \subset \mathbb{R}^d$ of m candidate centers, a penalty function p defined over the points of \mathcal{D} , where $p(j) \geq 0$ for each $j \in \mathcal{D}$, and an integer $k > 0$, the goal is to find a set $\mathcal{S} \subseteq \mathcal{F}$ of k centers such that the objective function $\sum_{j \in \mathcal{D}} \min\{c(j, \mathcal{S}), p(j)\}$ is minimized.*

We now show that a k -MPWP instance can be converted into a discrete k -MPWP instance with an arbitrarily small loss in the approximation ratio.

Lemma 3. *Given an instance $\mathcal{I}_1 = (\mathcal{D}, p, k)$ of the k -MPWP and a parameter $\epsilon > 0$, we can convert it into an instance $\mathcal{I}_2 = (\mathcal{D}, \mathcal{F}, p, k)$ of the discrete k -MPWP, such that any λ -approximation solution to \mathcal{I}_2 is a $\lambda(1+\epsilon)$ -approximation solution to \mathcal{I}_1 .*

Proof. Given an instance $\mathcal{I}_1 = (\mathcal{D}, p, k)$ of the k -MPWP, we compute an ϵ -approximate centroid set \mathcal{F} for \mathcal{D} and consider an instance $\mathcal{I}_2 = (\mathcal{D}, \mathcal{F}, p, k)$ of the discrete k -MPWP. Let \mathcal{S}_1^* and \mathcal{P}_1^* denote the sets of the opened centers and the penalized points in an optimal solution to \mathcal{I}_1 respectively. Let \mathcal{S}_2^* denote the set of the opened centers in an optimal solution to \mathcal{I}_2 . For each center $i \in \mathcal{S}_1^*$, let $\mathcal{D}_i = \{j \mid j \in \mathcal{D} \setminus \mathcal{P}_1^* \text{ and } \arg \min_{i' \in \mathcal{S}_1^*} c(j, i') = i\}$, and let $\tau(i) = \arg \min_{i' \in \mathcal{F}} \sum_{j \in \mathcal{D}_i} c(j, i')$. Define $\mathcal{S}' = \{\tau(i) \mid i \in \mathcal{S}_1^*\}$. Given a set $\mathcal{S} \subseteq \mathcal{F}$ of k centers that induces a λ -approximation to \mathcal{I}_2 , we have

$$\begin{aligned}
 \sum_{j \in \mathcal{D}} \min\{c(j, \mathcal{S}), p(j)\} &\leq \lambda \sum_{j \in \mathcal{D}} \min\{c(j, \mathcal{S}_2^*), p(j)\} \\
 &\leq \lambda \sum_{j \in \mathcal{D}} \min\{c(j, \mathcal{S}'), p(j)\} \\
 &\leq \lambda \left(\sum_{j \in \mathcal{D} \setminus \mathcal{P}_1^*} c(j, \mathcal{S}') + \sum_{j \in \mathcal{P}_1^*} p(j) \right) \\
 &\leq \lambda \left(\sum_{i \in \mathcal{S}_1^*} \sum_{j \in \mathcal{D}_i} c(j, \tau(i)) + \sum_{j \in \mathcal{P}_1^*} p(j) \right) \\
 &\leq \lambda(1 + \epsilon) \left(\sum_{i \in \mathcal{S}_1^*} \sum_{j \in \mathcal{D}_i} c(j, i) + \sum_{j \in \mathcal{P}_1^*} p(j) \right) \\
 &= \lambda(1 + \epsilon) \sum_{j \in \mathcal{D}} \min\{c(j, \mathcal{S}_1^*), p(j)\},
 \end{aligned}$$

where the fifth step follows from the definition of the approximate centroid set. □

Consider a discrete k -MPWP instance $(\mathcal{D}, \mathcal{F}, p, k)$, we have the following integer programming (IP) for the problem.

$$\min \quad \sum_{i \in \mathcal{F}, j \in \mathcal{D}} x_{ij} c(j, i) + \sum_{j \in \mathcal{D}} p(j) z_j \tag{IP1}$$

$$\text{s.t.} \quad \sum_{i \in \mathcal{F}} x_{ij} + z_j = 1 \quad \forall j \in \mathcal{D} \tag{1}$$

$$x_{ij} \leq y_i \quad \forall j \in \mathcal{D}, i \in \mathcal{F} \tag{2}$$

$$\sum_{i \in \mathcal{F}} y_i \leq k \tag{3}$$

$$x_{ij}, y_i, z_j \in \{0, 1\}. \quad \forall j \in \mathcal{D}, i \in \mathcal{F} \tag{4}$$

IP1 has a variable y_i for each $i \in \mathcal{F}$ that indicates whether i is opened in the solution, a variable z_j for each $j \in \mathcal{D}$ that indicates whether j is added to the penalized set, and a variable x_{ij} for each center-point pair $i \in \mathcal{F}$ and $j \in \mathcal{D}$ that indicates whether j is assigned to i . Constraint (1) enforces that each point must be either assigned a cluster center or added to the penalized set. Constraint (2) says that only the opened centers can be used to cluster the points. Constraint (3) enforces that only k centers can be opened. The cost of an optimal solution to IP1 is denoted by OPT_i .

3 The Algorithm

3.1 A Fractional Solution for the Discrete k -MPWP

We consider the following linear programming (LP) relaxation of IP1 where the integrality constraints on the variables are relaxed.

$$\min \sum_{i \in \mathcal{F}, j \in \mathcal{D}} x_{ij}c(j, i) + \sum_{j \in \mathcal{D}} p(j)z_j \tag{LP1}$$

$$\begin{aligned} \text{s.t.} \quad & (1), (2), \text{ and } (3) \\ & x_{ij}, y_i, z_j \geq 0. \quad \forall j \in \mathcal{D}, i \in \mathcal{F} \end{aligned} \tag{5}$$

The cost of an optimal solution to LP1 is denoted by OPT_f . Since LP1 is a relaxation of IP1, we have $OPT_f \leq OPT_i$.

For the discrete k -MPWP, the constraint on the number of the selected centers given in LP1 is one of the main obstacles. A commonly used way for overcoming such obstacle, as outlined in [17], is to consider the Lagrangian relaxation where we rule out the constraint but add the penalty for its violation to the objective function. This results in the following LP for any $\lambda \geq 0$.

$$\min \sum_{i \in \mathcal{F}, j \in \mathcal{D}} x_{ij}c(j, i) + \sum_{j \in \mathcal{D}} p(j)z_j + \lambda(\sum_{i \in \mathcal{F}} y_i - k) \tag{LP2(\lambda)}$$

$$\text{s.t.} \quad (1), (2), \text{ and } (5).$$

We can get integer solutions to LP2(λ) based on the primal-dual techniques introduced in [17] and [1]. Such solutions are almost feasible for LP1, except that constraint (3) may be violated. Given a solution $T' = (x', y', z')$ that satisfies constraints (1), (2), and (5), let $S(T') = \sum_{i \in \mathcal{F}, j \in \mathcal{D}} x'_{ij}c(j, i)$ and $P(T') = \sum_{j \in \mathcal{D}} p(j)z'_j$. The cost of T' on LP1 can be denoted by $V(T') = S(T') + P(T')$.

The following lemma shows that a convex combination of two integer solutions to LP2(λ) is a feasible fractional solution to LP1, whose cost is within a constant times OPT_f . The essential ideas of the lemma are based on standard techniques [1, 17]. The only difference to the previous primal-dual process is that we must deal with an additional constraint to enforce the points far from the cluster centers to pay their penalty costs.

Lemma 4. *Given a parameter $\epsilon > 0$, there is a polynomial time algorithm that yields two integer solutions $\mathcal{T}_1 = (x^1, y^1, z^1)$ and $\mathcal{T}_2 = (x^2, y^2, z^2)$ for LP2(λ), and the two solutions satisfy the following properties:*

1. $\sum_{i \in \mathcal{F}} y_i^1 = k_1 < k$ and $\sum_{i \in \mathcal{F}} y_i^2 = k_2 > k$;
2. Let $a = \frac{k_2 - k}{k_2 - k_1}$ and $b = 1 - a$. We have $aV(\mathcal{T}_1) + bV(\mathcal{T}_2) < (\eta + O(\epsilon))OPT_f$, where $\eta = 6.3575$.

3.2 Rounding

In this section, we give a deterministic rounding procedure which yields an integer solution to the discrete k -MPWP. Given an instance $(\mathcal{D}, \mathcal{F}, p, k)$ of the discrete k -MPWP and a parameter $\epsilon > 0$, we first use Lemma 4 to obtain two solutions \mathcal{T}_1 and \mathcal{T}_2 . Let \mathcal{S}_1 and \mathcal{S}_2 denote the sets of the opened centers in \mathcal{T}_1 and \mathcal{T}_2 respectively, where $|\mathcal{S}_1| = k_1 < k$ and $|\mathcal{S}_2| = k_2 > k$. Given a point $j \in \mathcal{D}$, let $i_1(j)$ denote its nearest point in \mathcal{S}_1 and $i_2(j)$ denote its nearest point in \mathcal{S}_2 . We say that j is captured by $i_1(j)$ and $i_2(j)$. Moreover, let \mathcal{P}_1 and \mathcal{P}_2 denote the penalized sets of the two solutions, i.e., $\mathcal{P}_1 = \{j \mid j \in \mathcal{D} \text{ and } z_j^1 = 1\}$ and $\mathcal{P}_2 = \{j \mid j \in \mathcal{D} \text{ and } z_j^2 = 1\}$.

We distinguish the analysis into the following two cases.

Case (1): $a \in (\frac{25}{78}, 1)$ or $V(\mathcal{T}_1) < \frac{78}{25}V(\mathcal{T}_2)$. Recall that $k_1 < k$ and thus \mathcal{T}_1 is feasible for LP1. In this case, we argue that $V(\mathcal{T}_1)$ should be close to OPT_i , and we can directly return \mathcal{T}_1 as a solution.

Case (2): $a \in (0, \frac{25}{78}]$ and $V(\mathcal{T}_1) \geq \frac{78}{25}V(\mathcal{T}_2)$. In this case, \mathcal{T}_1 might be quite expensive compared to the optimal solution. Our algorithm prefers to select the centers from \mathcal{S}_2 . We give different algorithms on the basis of the difference between k_1 and k . Specifically, we have the following two subcases.

Case (2.1): $k \leq k_1 + \frac{1}{\epsilon a}$. This captures the scenario that the difference between k_1 and k is upper-bounded by a small value. We argue that a subset of \mathcal{S}_2 of size k can be used as a solution and gives the desirable approximation ratio. We are able to show that such subset of \mathcal{S}_2 can be found in polynomial time.

Case (2.2): $k > k_1 + \frac{1}{\epsilon a}$. In this case, we start with using \mathcal{T}_1 as the solution to LP1, and then reduce the cost by replacing a portion of the center set with the centers from \mathcal{S}_2 . We achieve this by considering a Knapsack-type LP that maximizes the reduced cost, which is similar to the ones in [21] and [4] for the k -median problem. Note that the Knapsack-type LP in [21] and [4] opens $k + 2$ centers to get the approximation ratio for k -median clustering, which cannot be directly applied to solve the discrete k -MPWP. For the Knapsack-type LP applied in this paper, we open at most k centers to approximate the discrete k -MPWP, and the analysis given in [21] and [4] is not workable. Thus, we give a different analysis for the ratio of the discrete k -MPWP.

Case (1): $a \in (\frac{25}{78}, 1)$ or $V(\mathcal{T}_1) < \frac{78}{25}V(\mathcal{T}_2)$.

In this case, we argue that \mathcal{T}_1 gives the desired approximation to the optimal solution. We prove this by further considering the following three subcases.

Case (1.1): $V(\mathcal{T}_1) \leq V(\mathcal{T}_2)$. Lemma 4 implies that

$$V(\mathcal{T}_1) \leq aV(\mathcal{T}_1) + bV(\mathcal{T}_2) < (\eta + O(\epsilon))OPT_f \leq (\eta + O(\epsilon))OPT_i,$$

where $a = \frac{k_2 - k}{k_2 - k_1}$, $b = 1 - a$, and $\eta = 6.3575$.

Case (1.2): $V(\mathcal{T}_2) < V(\mathcal{T}_1) < \frac{78}{25}V(\mathcal{T}_2)$. We have

$$\begin{aligned} V(\mathcal{T}_1) &< \frac{78}{25}V(\mathcal{T}_2) < \frac{78}{25}(aV(\mathcal{T}_1) + bV(\mathcal{T}_2)) \\ &< \frac{78}{25}(\eta + O(\epsilon))OPT_f \leq \frac{78}{25}(\eta + O(\epsilon))OPT_i. \end{aligned}$$

Case (1.3): $V(\mathcal{T}_1) \geq \frac{78}{25}V(\mathcal{T}_2)$ and $a \in (\frac{25}{78}, 1)$. We have

$$\begin{aligned} V(\mathcal{T}_1) &\leq \frac{1}{a}(aV(\mathcal{T}_1) + bV(\mathcal{T}_2)) < \frac{\eta + O(\epsilon)}{a}OPT_f \\ &\leq \frac{\eta + O(\epsilon)}{a}OPT_i < \frac{78}{25}(\eta + O(\epsilon))OPT_i. \end{aligned}$$

Putting together, we get that \mathcal{T}_1 gives a $\frac{78}{25}(\eta + O(\epsilon))$ -approximation for the discrete k -MPWP.

Case (2.1): $a \in (0, \frac{25}{78}]$, $V(\mathcal{T}_1) \geq \frac{78}{25}V(\mathcal{T}_2)$, and $k \leq k_1 + \frac{1}{\epsilon a}$.

In this subcase, we show that \mathcal{S}_2 contains k cluster centers that give a desired approximation for the discrete k -MPWP. Our algorithm for this subcase enumerates all the subsets of \mathcal{S}_2 of size k and return the one with the minimum cost on the problem. We first show that this enumeration can be completed quickly. We achieve this by proving that the difference between $|\mathcal{S}_2|$ and k is small.

Lemma 5. $k_2 - k \leq \frac{78}{53\epsilon}$.

Proof. Observe that $k_2 - k = (k - k_1)\frac{a}{1-a} \leq \frac{1}{\epsilon(1-a)} \leq \frac{78}{53\epsilon}$, where the first step follows from the fact that $a = \frac{k_2 - k}{k_2 - k_1}$, the second step follows from the assumption that $k \leq k_1 + \frac{1}{\epsilon a}$, and the last step follows from the assumption that $a \leq \frac{25}{78}$. \square

Lemma 5 implies that our algorithm considers at most $k^{O(1/\epsilon)}$ subsets of \mathcal{S}_2 . Thus, the algorithm can be executed in time $O(ndk^{O(1/\epsilon)})$, where $n = |\mathcal{D}|$. We proceed by proving that the set of centers given by the algorithm achieves the desired approximation ratio.

Lemma 6. *There exists a set $\mathcal{S}' \subset \mathcal{S}_2$, such that $|\mathcal{S}'| = k$ and $\sum_{j \in \mathcal{D}} \min\{c(j, \mathcal{S}'), p(j)\} < (3 + 2\eta + O(\epsilon))OPT_i$.*

Lemma 6 implies that our algorithm achieves a $(3 + 2\eta + O(\epsilon))$ -approximation for the discrete k -MPWP in case (2.1).

Case (2.2): $a \in (0, \frac{25}{78}]$, $V(\mathcal{T}_1) \geq \frac{78}{25}V(\mathcal{T}_2)$, and $k > k_1 + \frac{1}{\epsilon a}$.

In this subcase, we combine \mathcal{T}_1 and \mathcal{T}_2 into the required approximation solution. We achieve this by considering \mathcal{T}_1 as the solution initially, and then reducing the cost by replacing a portion of the center set with the centers from \mathcal{S}_2 by a greedy strategy.

Our algorithm is based on a notion of stars. For each point $i' \in \mathcal{S}_2$, let $\pi(i') \in \mathcal{S}_1$ be the nearest point of i' in \mathcal{S}_1 . For each $i \in \mathcal{S}_1$, we view i as the central point of a star. Let $\mathcal{L}_i = \{i' \mid i' \in \mathcal{S}_2 \text{ and } \pi(i') = i\}$. This is the set of leaves of the star with central point i . Given a point $j \in \mathcal{D}$, let $f_1(j) = \min\{c(j, i_1(j)), p(j)\}$ and $f_2(j) = \min\{c(j, i_2(j)), p(j)\}$. We first show that $\min\{c(j, \pi(i_2(j))), p(j)\}$ is bounded by a combination of $f_1(j)$ and $f_2(j)$.

Lemma 7. *For each point $j \in \mathcal{D}$, $\min\{c(j, \pi(i_2(j))), p(j)\} \leq 8f_2(j) + 2f_1(j)$.*

Proof. Given a point $j \in \mathcal{D}$, if j is penalized in \mathcal{T}_1 or \mathcal{T}_2 , then we have $p(j) \leq 8f_2(j) + 2f_1(j)$, as desired. If j is not penalized in \mathcal{T}_1 or \mathcal{T}_2 , we have

$$\begin{aligned} d(j, \pi(i_2(j))) &\leq d(j, i_2(j)) + d(i_2(j), \pi(i_2(j))) \\ &\leq d(j, i_2(j)) + d(i_2(j), i_1(j)) \\ &\leq d(j, i_2(j)) + d(j, i_2(j)) + d(j, i_1(j)) \\ &= 2d(j, i_2(j)) + d(j, i_1(j)), \end{aligned}$$

where the first and the third steps are due to triangle inequality, and the second step follows from the fact that $\pi(i_2(j))$ is the nearest point to $i_2(j)$ in \mathcal{S}_1 . This implies that

$$\begin{aligned} c(j, \pi(i_2(j))) &\leq (2d(j, i_2(j)) + d(j, i_1(j)))^2 \\ &\leq 8c(j, i_2(j)) + 2c(j, i_1(j)) \\ &= 8f_2(j) + 2f_1(j), \end{aligned}$$

where the second step follows from the fact that for any two values u and v , $(u + v)^2 = u^2 + 2uv + v^2 \leq 2u^2 + 2v^2$. This completes the proof of Lemma 7. \square

The idea of our algorithm is to ensure that for each star, either its central point or all its leaves are opened in the final solution, such that each point j can always be assigned to either $\pi(i_2(j))$ or $i_2(j)$. This implies that the cost induced by j is at most $f_2(j)$ if $i_2(j)$ is opened. Otherwise the cost induced by j is no more than $8f_2(j) + 2f_1(j)$ by Lemma 7. Consider a simple solution that opens the central points of all the stars. The cost of such solution is upper-bounded by $\sum_{j \in \mathcal{D}} 8f_2(j) + 2f_1(j) = 8V(\mathcal{T}_2) + 2V(\mathcal{T}_1)$. We can reduce the cost via opening the leaves of some stars. For a star with central point i , if we close i and open all the leaves of the star, we can then reduce the cost by $\sum_{j \in \sigma(\mathcal{L}_i)} 7f_2(j) + 2f_1(j)$, where $\sigma(\mathcal{L}_i)$ denotes the set of the points captured by a center in \mathcal{L}_i . We consider the following LP that maximizes the reduction in the cost.

$$\max \sum_{i \in \mathcal{S}_1} \sum_{j \in \sigma(\mathcal{L}_i)} (7f_2(j) + 2f_1(j))x_i \tag{LP3}$$

$$\text{s.t. } \sum_{i \in \mathcal{S}_1} x_i (|\mathcal{L}_i| - 1) \leq k - k_1 \tag{6}$$

$$0 \leq x_i \leq 1. \quad \forall i \in \mathcal{S}_1 \tag{7}$$

This LP has a variable x_i for each $i \in \mathcal{S}_1$. We open all the centers in \mathcal{L}_i if $x_i = 1$ and open i if $x_i = 0$. x_i taking value 1 increases the number of the opened centers by $|\mathcal{L}_i| - 1$. Constraint (6) enforces that at most k centers can be opened.

Let $\mathcal{X}^* = (x_1^*, \dots, x_{k_1}^*)$ denote an optimal solution to LP3. We know that this solution has at most one fractional variable since LP3 is a Knapsack LP. Let \mathcal{C}_1 denote the set of the center $i \in \mathcal{S}_1$ with $x_i^* = 1$ and \mathcal{C}_0 denote the set of the center $i \in \mathcal{S}_1$ with $x_i^* = 0$. Let l denote the center associated with the fractional variable. Define opt as the value of \mathcal{X}^* on LP3. Our algorithm for case (2.2) opens all the centers in \mathcal{C}_0 and the corresponding leaves of the centers in \mathcal{C}_1 . For center l , the algorithm opens l and a set of $\lceil x_l^* |\mathcal{L}_l| \rceil - 1$ centers in \mathcal{L}_l that maximizes the reduction in the cost. Let \mathcal{S} denote the set of the centers opened by the algorithm.

We show that our algorithm opens at most k centers.

Lemma 8. $|\mathcal{S}| \leq k$.

Lemma 8 implies that the set \mathcal{S} of centers is feasible for the discrete k -MPWP. We now consider the approximation ratio \mathcal{S} achieves. We first prove that the reduction in the cost induced by \mathcal{S} , called R , is close to opt .

Lemma 9. $R > (1 - a\epsilon)opt$.

Let $\text{cost}(\mathcal{S}) = \sum_{j \in \mathcal{D}} \min\{c(j, \mathcal{S}), p(j)\}$ denote the cost of \mathcal{S} on the discrete k -MPWP. We now show that \mathcal{S} achieves the desired approximation for the problem.

Lemma 10. $\text{cost}(\mathcal{S}) < (\frac{409}{131}\eta + O(\epsilon))OPT_i$.

Putting everything together, the approximation ratio of our algorithm on the discrete k -MPWP is upper-bounded by $\max\{\frac{78}{25}\eta, 3 + 2\eta, \frac{409}{131}\eta\} + O(\epsilon)$, which is no more than $19.849 + O(\epsilon)$ by the fact that $\eta = 6.3575$. Using Lemma 3, the algorithm also induces a $(19.849 + O(\epsilon))$ -approximation for the standard k -MPWP.

References

1. Ahmadian, S., Norouzi-Fard, A., Svensson, O., Ward, J.: Better guarantees for k -means and Euclidean k -median by primal-dual algorithms. In: Proceedings of 58th IEEE Symposium on Foundations of Computer Science, pp. 61–72 (2017)
2. Aloise, D., Deshpande, A., Hansen, P., Popat, P.: NP-hardness of Euclidean sum-of-squares clustering. *Mach. Learn.* **75**(2), 245–248 (2009)
3. Arthur, D., Vassilvitskii, S.: k -means++: the advantages of careful seeding. In: Proceedings of 18th ACM-SIAM Symposium on Discrete Algorithms, pp. 1027–1035 (2007)

4. Byrka, J., Pensyl, T., Rybicki, B., Srinivasan, A., Trinh, K.: An improved approximation for k -median and positive correlation in budgeted optimization. *ACM Trans. Algorithms* **13**(2), 23 (2017)
5. Charikar, M., Khuller, S., Mount, D.M., Narasimhan, G.: Algorithms for facility location problems with outliers. In: *Proceedings of 12th ACM-SIAM Symposium on Discrete Algorithms*, pp. 642–651 (2001)
6. Chen, K.: A constant factor approximation algorithm for k -median clustering with outliers. In: *Proceedings of 19th ACM-SIAM Symposium on Discrete Algorithms*, pp. 826–835 (2008)
7. Cohen-Addad, V., Klein, P.N., Mathieu, C.: Local search yields approximation schemes for k -means and k -median in Euclidean and minor-free metrics. In: *Proceedings of 57th IEEE Symposium on Foundations of Computer Science*, pp. 353–364 (2016)
8. Feldman, D., Schulman, L.J.: Data reduction for weighted and outlier-resistant clustering. In: *Proceedings of 23rd ACM-SIAM Symposium on Discrete Algorithms*, pp. 1343–1354 (2012)
9. Friggstad, Z., Khodamoradi, K., Rezapour, M., Salavatipour, M.R.: Approximation schemes for clustering with outliers. In: *Proceedings of 28th ACM-SIAM Symposium on Discrete Algorithms*, pp. 398–414 (2018)
10. Friggstad, Z., Rezapour, M., Salavatipour, M.R.: Local search yields a PTAS for k -means in doubling metrics. In: *Proceedings of 57th IEEE Symposium on Foundations of Computer Science*, pp. 365–374 (2016)
11. Guha, S., Li, Y., Zhang, Q.: Distributed partial clustering. In: *Proceedings of 29th ACM Symposium on Parallelism in Algorithms and Architectures*, pp. 143–152 (2017)
12. Gupta, A., Guruganesh, G., Schmidt, M.: Approximation algorithms for aversion k -clustering via local k -median. In: *Proceedings of 43rd International Colloquium on Automata, Languages and Programming*, pp. 1–13 (2016)
13. Gupta, S., Kumar, R., Lu, K., Moseley, B., Vassilvitskii, S.: Local search methods for k -means with outliers. *Proc. VLDB Endow.* **10**(7), 757–768 (2017)
14. Hajiaghayi, M., Khandekar, R., Kortsarz, G.: Local search algorithms for the red-blue median problem. *Algorithmica* **63**(4), 795–814 (2012)
15. Huang, L., Jiang, S., Li, J., Wu, X.: ϵ -coresets for clustering (with outliers) in doubling metrics. In: *Proceedings of 50th ACM Symposium on Theory of Computing*, pp. 814–825 (2018)
16. Jain, K., Mahdian, M., Markakis, E., Saberi, A., Vazirani, V.V.: Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. *J. ACM* **50**(6), 795–824 (2003)
17. Jain, K., Vazirani, V.V.: Approximation algorithms for metric facility location and k -median problems using the primal-dual schema and lagrangian relaxation. *J. ACM* **48**(2), 274–296 (2001)
18. Kanungo, T., Mount, D.M., Netanyahu, N.S., Piatko, C.D., Silverman, R., Wu, A.Y.: A local search approximation algorithm for k -means clustering. *Comput. Geom.* **28**(2–3), 89–112 (2004)
19. Kumar, A., Sabharwal, Y., Sen, S.: Linear-time approximation schemes for clustering problems in any dimensions. *J. ACM* **57**(2), 1–32 (2010)
20. Li, S., Guo, X.: Distributed k -clustering for data with heavy noise. In: *Proceedings of 32nd Annual Conference on Neural Information Processing Systems*, pp. 7849–7857 (2018)
21. Li, S., Svensson, O.: Approximating k -median via pseudo-approximation. *SIAM J. Comput.* **45**(2), 530–547 (2016)

22. Li, Y., Du, D., Xiu, N., Xu, D.: Improved approximation algorithms for the facility location problems with linear/submodular penalties. *Algorithmica* **73**(2), 460–482 (2015)
23. Mahajan, M., Nimbhorkar, P., Varadarajan, K.: The planar k -means problem is NP-hard. *Theoret. Comput. Sci.* **442**, 13–21 (2012)
24. Makarychev, K., Makarychev, Y., Sviridenko, M., Ward, J.: A bi-criteria approximation algorithm for k -means. In: Proceedings of 19th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems and 20th International Workshop on Randomization and Computation, pp. 1–20 (2016)
25. Matousek, J.: On approximate geometric k -clustering. *Discrete Comput. Geom.* **24**(1), 61–84 (2000)
26. Ravishankar, K., Li, S., Sai, S.: Constant approximation for k -median and k -means with outliers via iterative rounding. In: Proceedings of 50th ACM Symposium on Theory of Computing, pp. 646–659 (2018)
27. Wu, C., Du, D., Xu, D.: An approximation algorithm for the k -median problem with uniform penalties via pseudo-solution. *Theoret. Comput. Sci.* **749**, 80–92 (2018)
28. Xu, G., Xu, J.: An LP rounding algorithm for approximating uncapacitated facility location problem with penalties. *Inf. Process. Lett.* **94**(3), 119–123 (2005)
29. Xu, G., Xu, J.: An improved approximation algorithm for uncapacitated facility location problem with penalties. *J. Comb. Optim.* **17**(4), 424–436 (2009)
30. Zhang, D., Hao, C., Wu, C., Xu, D., Zhang, Z.: A local search approximation algorithm for the k -means problem with penalties. In: Cao, Y., Chen, J. (eds.) COCOON 2017. LNCS, vol. 10392, pp. 568–574. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-62389-4_47

Author Index

- Asahiro, Yuichi 38
Ashok, Pradeesha 26
- Bhagat, Subhash 144
- Cai, Qingqiong 111
Cao, Yixin 51
Chang, Jou-Ming 88
Chang, Ruay-Shiung 88
Chen, Yong 14
Cheng, Yukun 76
- Deng, Xiaotie 76
- Feng, Qilong 170
- Gasieniec, Leszek 156
Goebel, Randy 14
- Huang, Shenwei 111
- Jansson, Jesper 38, 156
Jiang, Zhihao 121
- Kleine Büning, Hans 100
- Lai, Wenxing 133
Levcopoulos, Christos 156
Li, Tao 111
Lin, Guohui 14
Lingas, Andrzej 156
Liu, Longcheng 14
- Miyano, Eiji 38
Mizuki, Takaaki 63
Mukhopadhyaya, Krishnendu 144
- Ono, Hiroataka 38
- Pai, Kung-Jui 88
Persson, Mia 156
- Reddy, Meghana M. 26
- Shi, Feng 170
Shi, Yongtang 111
Shinagawa, Kazumasa 63
Su, Bing 14
Subramani, K. 100
- T. P., Sandhya 38
Tong, Weitian 14
- Wang, Jianxin 51, 170
Wojciechowski, P. 100
Wu, Ro-Yu 88
- Xu, Yao 14
- Yan, Xiang 1
You, Jie 51
- Zhang, An 14
Zhang, Mengqian 76
Zhang, Zhen 170
Zhao, Haoyu 121
Zhu, Wei 1