

Ding-Zhu Du  
Donglei Du  
Chenchen Wu  
Dachuan Xu (Eds.)

LNCS 13571

# Theory and Applications of Models of Computation

17th Annual Conference, TAMC 2022  
Tianjin, China, September 16–18, 2022  
Proceedings

T			2
	A	2	
	0	M	
2			C

## Founding Editors

Gerhard Goos

*Karlsruhe Institute of Technology, Karlsruhe, Germany*

Juris Hartmanis

*Cornell University, Ithaca, NY, USA*


## Editorial Board Members

Elisa Bertino

*Purdue University, West Lafayette, IN, USA*

Wen Gao

*Peking University, Beijing, China*

Bernhard Steffen 

*TU Dortmund University, Dortmund, Germany*

Moti Yung 

*Columbia University, New York, NY, USA*


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


Ding-Zhu Du · Donglei Du · Chenchen Wu ·  
Dachuan Xu (Eds.)

# Theory and Applications of Models of Computation

17th Annual Conference, TAMC 2022  
Tianjin, China, September 16–18, 2022  
Proceedings

*Editors*

Ding-Zhu Du   
University of Texas at Dallas  
Richardson, TX, USA

Donglei Du   
University of New Brunswick  
Fredericton, NB, Canada

Chenchen Wu   
Tianjin University of Technology  
Tianjin, China

Dachuan Xu  
Beijing University of Technology  
Beijing, China

ISSN 0302-9743                      ISSN 1611-3349 (electronic)  
Lecture Notes in Computer Science  
ISBN 978-3-031-20349-7              ISBN 978-3-031-20350-3 (eBook)  
<https://doi.org/10.1007/978-3-031-20350-3>

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

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

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

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

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

# Preface

The 17th Annual Conference on Theory and Applications of Models of Computation (TAMC 2022) took place in Tianjin, China, during September 16–18, 2022. TAMC 2022 provided an excellent venue for researchers in the area of computational theory, information theory, and applications. The main themes of the conference were computability, complexity, algorithms, information theory, and their extensions to machine learning theory, and foundations of artificial intelligence.

The Program Committee received a total of 75 submissions, among which 33 were accepted for presentation at the conference. Each contributed paper was subject to a rigorous single-blind peer review process, with reviewers drawn from a large group of members of the Program Committee. On average, each paper received 3 reviews.

We would like to express our sincere appreciation to everyone who made TAMC 2022 a success by volunteering their time and effort: the authors, the Program Committee members, and the reviewers. We thank Springer for accepting TAMC 2022 for publication in this Lecture Notes in Computer Science (LNCS) proceedings volume. Our special thanks also extend to the other chairs and Organizing Committee members for their excellent work.

August 2022

Ding-Zhu Du  
Donglei Du  
Chenchen Wu  
Dachuan Xu

# Organization

## Steering Committee

Agrawal, Manindra	Indian Institute of Technology Kanpur, India
Cai, Jin-Yi	University of Wisconsin-Madison, USA
Hopcroft, John	Cornell University, USA
Li, Angsheng	Beihang University, China
Liu, Zhiyong	Institute of Computing Technology, Chinese Academy of Sciences, China

## General Chair

Du, Ding-Zhu	University of Texas at Dallas, USA
--------------	------------------------------------

## Program Committee Chairs

Du, Donglei	University of New Brunswick, Canada
Wu, Chenchen	Tianjin University of Technology, China
Xu, Dachuan	Beijing University of Technology, China

## Local Organizing Chairs

Huang, Xujian	Tianjin University of Technology, China
Liu, Xin-Wei	Hebei University of Technology, China
Shi, Yongtang	Nankai University, China

## Finance Chairs

Lei, Hui	Nankai University, China
Xu, Yicheng	Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, China

## Publication Chairs

Han, Lu	Beijing University of Posts and Telecommunications, China
Yang, Ruiqi	Beijing University of Technology, China

## Web Chairs

Ji, Sai

Zhong, Xinxin

Hebei University of Technology, China

Tianjin University of Technology, China

## Registration Chairs

Sun, Jian

Wang, Yijing

Nankai University, China

Academy of Mathematics and Systems Science,  
Chinese Academy of Sciences, China

## Program Committee

Bampis, Evripidis

Bonato, Anthony

Chau, Vincent

Chen, Xujin

Ding, Hu

Feng, Qilong

Guo, Longkun

Hong, Seok-Hee

Kling, Peter

Lempp, Steffen

Letsios, Dimitrios

Li, Minming

Li, Jian

Lucarelli, Giorgio

Pavan, A.

Seth, Anil

Shi, Yangguang

Sun, Xiaoming

Sorbonne University, France

Toronto Metropolitan University, Canada

Southeast University, China

Academy of Mathematics and Systems Science,  
Chinese Academy of Sciences, China

University of Science and Technology of China,  
China

Central South University, China

Fuzhou University, China

University of Sydney, Australia

University of Hamburg, Germany

University of Wisconsin-Madison, USA

King's College London, UK

City University of Hong Kong, China

Tsinghua University, China

University of Lorraine, France

Iowa State University, USA

Indian Institute of Technology Kanpur, India

Shandong University, China

Institute of Computing Technology, Chinese  
Academy of Sciences, China

Paris-Saclay University, France

University of Utah, USA

Lafayette College, USA

Institute of Software, Chinese Academy of  
Sciences, China

State University of New York at Buffalo, USA

University of Patras, Greece

University of Alabama in Huntsville, USA

Shandong University, China

Thang, Nguyen Kim

Wang, Haitao

Xia, Ge

Xia, Mingji

Xu, Jinhui

Zaroliagis, Christos

Zhang, Huaming

Zhang, Peng



Zhang, Xiaoyan

Zhang, Yong

Zhang, Zhao

Zissimopoulos, Vassilis

Nanjing Normal University, China

Shenzhen Institute of Advanced Technology,

Chinese Academy of Sciences, China

Zhejiang Normal University, China

National and Kapodistrian University of Athens,  
Greece

## **Additional Reviewers**

Basu, Samik

Chen, Yong

Dixon, Peter

Guo, Xinru

Huang, Jiawei

Kononov, Alexander

Lamprou, Ioannis

Li, Shimin

Lian, Yuefang

Liu, Pengcheng

Minich, Sarah

Ran, Yingli

Sigalas, Ioannis

Soullignac, Francisco

Stull, Don

Sun, Xin

Vaxevanakis, Ioannis

Wang, Kaiyun

Yu, Wei

Zhang, Yapu

Zhao, Yiming

Zhao, Yingchao

Zhou, Xiangnan

# Contents

Maximization of $k$ -Submodular Function with a Matroid Constraint .....	1
<i>Yunjing Sun, Yuezhu Liu, and Min Li</i>	
Maximizing Approximately Non- $k$ -Submodular Monotone Set Function with Matroid Constraint .....	11
<i>YanJun Jiang, Yijing Wang, Ruiqi Yang, and Weina Ye</i>	
Time-of-Use Scheduling Problem with Equal-Length Jobs .....	21
<i>Vincent Chau, Chenchen Fu, Yan Lyu, Weiwei Wu, and Yizheng Zhang</i>	
Online Weakly DR-Submodular Optimization with Stochastic Long-Term Constraints .....	32
<i>Junkai Feng, Ruiqi Yang, Yapu Zhang, and Zhenning Zhang</i>	
Physical ZKP for Makaro Using a Standard Deck of Cards .....	43
<i>Suthee Ruangwises and Toshiya Itoh</i>	
Characterization of the Imbalance Problem on Complete Bipartite Graphs .....	55
<i>Steven Ge and Toshiya Itoh</i>	
New Algorithms for a Simple Measure of Network Partitioning .....	67
<i>Xueyang Zhao, Binghao Yan, and Peng Zhang</i>	
On Two Types of Concept Lattices in the Theory of Numberings .....	79
<i>Nikolay Bazhenov, Manat Mustafa, and Anvar Nurakunov</i>	
Computing Connected- $k$ -Subgraph Cover with Connectivity Requirement .....	93
<i>Pengcheng Liu, Zhao Zhang, Yingli Ran, and Xiaohui Huang</i>	
Analyzing the 3-path Vertex Cover Problem in Planar Bipartite Graphs .....	103
<i>Sangram K. Jena and K. Subramani</i>	
Competition-Based Generalized Self-profit Maximization in Dual-Attribute Networks .....	116
<i>Liman Du, Wenguo Yang, and Suixiang Gao</i>	
Largest Convex Hulls for Constant Size, Convex-Hull Disjoint Clusters .....	128
<i>Xuehou Tan and Rong Chen</i>	

Two-Stage Submodular Maximization Under Knapsack and Matroid Constraints .....	140
<i>Zhicheng Liu, Jing Jin, Donglei Du, and Xiaoyan Zhang</i>	
$(\mathbb{Z}, \text{succ}, U)$ , $(\mathbb{Z}, E, U)$ , and Their CSP's .....	155
<i>William Gasarch, Michael Laskowski, and Shaopeng Zhu</i>	
Circle Graph Isomorphism in Almost Linear Time .....	176
<i>Vít Kalisz, Pavel Klavík, and Peter Zeman</i>	
The Exact Subset MultiCover Problem .....	189
<i>Emile Benoist, Guillaume Fertin, and Géraldine Jean</i>	
Hide a Liar: Card-Based ZKP Protocol for Usowan .....	201
<i>Léo Robert, Daiki Miyahara, Pascal Lafourcade, and Takaaki Mizuki</i>	
Complexity Analysis of a Stochastic Variant of Generalized Alternating Direction Method of Multipliers .....	218
<i>Jia Hu, Tiande Guo, and Congying Han</i>	
A $3/4$ Differential Approximation Algorithm for Traveling Salesman Problem .....	237
<i>Yuki Amano and Kazuhisa Makino</i>	
Exact and Parameterized Algorithms for Restricted Subset Feedback Vertex Set in Chordal Graphs .....	249
<i>Tian Bai and Mingyu Xiao</i>	
An Approximation Algorithm for the $B$ -prize-collecting Multicut Problem in Trees .....	262
<i>Xiaofei Liu and Weidong Li</i>	
Two-Stage Non-submodular Maximization .....	272
<i>Hong Chang, Zhicheng Liu, Ping Li, and Xiaoyan Zhang</i>	
Fault-Tolerant Total Domination via Submodular Function Approximation .....	281
<i>Ioannis Lamprou, Ioannis Sigalas, Ioannis Vaxevanakis, and Vassilis Zissimopoulos</i>	
On the Parallel Complexity of Constrained Read-Once Refutations in UTVPI Constraint Systems .....	293
<i>K. Subramani and Piotr Wojciechowski</i>	
Extracting Densest Sub-hypergraph with Convex Edge-Weight Functions .....	305
<i>Yi Zhou, Shan Hu, and Zimo Sheng</i>	

Exact and Approximation Algorithms for PMMS Under Identical Constraints ..... 322  
*Sijia Dai, Guichen Gao, Xinru Guo, and Yong Zhang*

Finite-State Relative Dimension, Dimensions of AP Subsequences and a Finite-State van Lambalgen’s Theorem ..... 334  
*Satyadev Nandakumar, Subin Pulari, and Akhil S*

Distributed Connected Dominating Sets in Unit Square and Disk Graphs ..... 346  
*Barun Gorain, Kaushik Mondal, and Supantha Pandit*

An Inventory System Optimization for Solving Joint Pricing and Ordering Problem with Trapezoidal Demand and Partial Backlogged Shortages in a Limited Sales Period ..... 359  
*Chunming Xu, Mingfei Bai, Qiyue Wang, and Yiwei Wang*

A Set-Theoretic Representation of Algebraic L-domains ..... 370  
*Juan Zou, Yuhan Zhao, Cuixia Miao, and Longchun Wang*

Normality, Randomness and Kolmogorov Complexity of Continued Fractions ..... 382  
*Prateek Vishnoi*

Weakly  $k$ -submodular Maximization Under Matroid Constraint ..... 393  
*Yijing Wang, Dongmei Zhang, Yapu Zhang, and Zhenning Zhang*

Approximation Algorithms for Diversity-Bounded Center Problems ..... 402  
*Lu Han, Shuilian Liu, Yicheng Xu, and Yong Zhang*

**Author Index** ..... 415



# Maximization of $k$ -Submodular Function with a Matroid Constraint

Yunjing Sun, Yuezhu Liu, and Min Li<sup>(✉)</sup>

School of Mathematics and Statistics, Shandong Normal University, Jinan 250014,  
People's Republic of China  
liminemily@sdnu.edu.cn

**Abstract.** A  $k$ -submodular function is a promotion of a submodular function, whose domain is composed of  $k$  disjoint subsets rather than a single subset. In this paper, we give a deterministic algorithm for the non-monotone  $k$ -submodular function maximization problem subject to a matroid constraint with approximation factor  $1/3$ . Based on this result, we give a randomized  $1/3$ -approximation algorithm for the problem with faster running time, but the probability of success is  $(1 - \varepsilon)$ . And we obtain that the complexity of deterministic algorithm and random algorithm is  $O(N|D|(p + kq))$  and  $O(|D|(p \log \frac{N}{\varepsilon_1} + kq \log \frac{N}{\varepsilon_2}) \log N)$  respectively, where  $D$  is the ground set of the matroid constraint with rank  $N$ ,  $p$  is times of oracle to calculate whether a set is an independent set in this matroid,  $q$  is the times of oracle to calculate a value of the  $k$ -submodular function, and  $\varepsilon, \varepsilon_1, \varepsilon_2$  are positive parameters with  $\varepsilon = \max\{\varepsilon_1, \varepsilon_2\}$ .

**Keywords:**  $k$ -submodular function · Matroid constraint · Deterministic algorithm · Randomized algorithm

## 1 Introduction

The definition of  $k$ -submodular function is generalized from submodular case. For non-monotone unconstrained  $k$ -submodular functions maximization, Ward and Živný give a  $1/(1 + a)$ -approximation algorithm, where  $a = \max\{1, \sqrt{(k - 1)/4}\}$  [14]. Iwata et al. [3] design a new randomized algorithm and get the approximation ratio of  $1/2$ . Then, Oshima [6] shows another improved randomized  $\frac{k^2+1}{2k^2+1}$ -approximation algorithm for  $k \geq 3$  and a random  $\frac{\sqrt{17}-3}{2}$ -approximation algorithm for  $k = 3$ . For monotone unconstrained case, a  $\frac{k}{2k-1}$ -approximation algorithm is proposed in [3]. There are also some works about the monotone constrained cases. In order to maximize  $k$ -submodular function with the total size constraint, Ohsaka and Yoshida [5] propose a constant-factor approximation algorithm, giving an approximation ratio of  $1/2$ . In addition, they also propose an almost linear-time algorithm by random sampling and get  $1/2$ -approximation algorithm with probability of  $(1 - \delta)$ . Meanwhile, they present a  $1/3$ -approximation algorithm for the individual size constraints. A knapsack constraint is also studied for the

---

Supported by Natural Science Foundation of Shandong Province (No. ZR2020MA029) of China.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022  
D.-Z. Du et al. (Eds.): TAMC 2022, LNCS 13571, pp. 1–10, 2022.  
[https://doi.org/10.1007/978-3-031-20350-3\\_1](https://doi.org/10.1007/978-3-031-20350-3_1)

constraint problems. With this kind of constraint, Tang et al. get a deterministic  $(\frac{1}{2} - \frac{1}{2e})$ -approximation algorithm [12]. Later, they make a further streaming research for cardinality and knapsack constraints [11]. Recently, for monotone  $k$ -submodular function maximization with a matroid constraint, Sakaue [10] gives a greedy algorithm that outputs an approximation ratio of  $1/2$ . On this basis, Rafiey and Yoshida [9] design a random algorithm to reduce the complexity. Besides, Wang and Zhou [13] provide a new method called multilinear extension for  $k$ -submodular maximization problem. Qian et al. [8] also propose a new way to maximize the given objective and minimize the size concurrently by using a multiobjective evolutionary algorithm. A novel budgeted  $k$ -submodular maximization problem with streaming algorithms is researched in [7]. There are also some works for  $k$ -submodular minimization [1, 2].

The structure of this article is given as follows. We briefly introduce the relevant knowledge and conclusions about  $k$ -submodular function and matroids in Sect. 2. In Sect. 3, we design a deterministic algorithm with the complexity of  $O(N|D|(p + kq))$  and prove the  $1/3$ -approximation ratio for non-monotone  $k$ -submodular function maximization under a matroid constraint. Then, we design a random  $1/3$ -approximation algorithm with probability of  $(1 - \varepsilon)$ , and get the complexity of  $O(|D|(p \log \frac{N}{\varepsilon_1} + kq \log \frac{N}{\varepsilon_2}) \log N)$  by this algorithm, where  $\varepsilon = \max\{\varepsilon_1, \varepsilon_2\}$ .

## 2 Preliminaries

First of all, we will introduce the related content of  $k$ -submodular functions used in this paper. Given a finite nonempty set  $D$ ,  $2^D$  is the power set of  $D$ . A set function  $f: 2^D \rightarrow \mathbb{R}$  defined on power set of  $D$  is called *submodular* if for all  $U, V \subseteq D$ ,

$$f(U) + f(V) \geq f(U \cap V) + f(U \cup V).$$

We define the marginal value of  $e$  with respect to  $U$  as  $f_U(e) = f(U \cup \{e\}) - f(U)$  for all  $U \subseteq D$  and  $e \notin U$ . Then we have the property of diminishing marginal gain of a submodular function:

$$f(U \cup \{e\}) - f(U) \geq f(V \cup \{e\}) - f(V),$$

for any  $U \subseteq V$  and  $e \in D \setminus V$ .

$k$ -submodular function has  $k + 1$  choices for the placement of elements, rather than just two choices. Given a positive integer  $k \geq 1$  and a finite nonempty set  $D$ , let  $(k + 1)^D := \{(U_1, \dots, U_k) \mid U_l \subseteq D, \forall l \in \{1, 2, \dots, k\}, U_l \cap U_n = \emptyset, \forall l \neq n\}$ . A function  $f: (k + 1)^D \rightarrow \mathbb{R}$  is called  *$k$ -submodular*, if for any  $\mathbf{u} = (U_1, \dots, U_k)$  and  $\mathbf{v} = (V_1, \dots, V_k) \in (k + 1)^D$ , we have

$$f(\mathbf{u}) + f(\mathbf{v}) \geq f(\mathbf{u} \sqcup \mathbf{v}) + f(\mathbf{u} \sqcap \mathbf{v}),$$

where

$$\begin{aligned} \mathbf{u} \sqcap \mathbf{v} &:= (U_1 \cap V_1, \dots, U_k \cap V_k), \\ \mathbf{u} \sqcup \mathbf{v} &:= \left( U_1 \cup V_1 \setminus \left( \bigcup_{l \neq 1} (U_l \cup V_l) \right), \dots, U_k \cup V_k \setminus \left( \bigcup_{l \neq k} (U_l \cup V_l) \right) \right). \end{aligned}$$

For any  $\mathbf{u} = (U_1, \dots, U_k) \in (k+1)^D$ , we denote the support of  $\mathbf{u}$  by  $\text{supp}(\mathbf{u}) = \cup_{l=1}^k U_l$  and call the cardinality of  $\text{supp}(\mathbf{u})$  as the size of  $\mathbf{u}$ . And for any element  $e \in D$ , we denote

$$\mathbf{u}(e) = \begin{cases} l, & \text{if } e \in U_l, \\ 0, & \text{if } e \notin \text{supp}(\mathbf{u}). \end{cases}$$

A partial ordering relation is defined as follows, for  $\mathbf{u} = (U_1, \dots, U_k)$  and  $\mathbf{v} = (V_1, \dots, V_k) \in (k+1)^D$ ,  $\mathbf{u} \preceq \mathbf{v}$  if  $U_l \subseteq V_l$  for all  $l \in [k]$ , where  $[k] := \{1, 2, 3, \dots, k\}$ . Then a  $k$ -submodular function  $f$  is *monotone* if  $f(\mathbf{u}) \leq f(\mathbf{v})$  for any  $\mathbf{u} \preceq \mathbf{v}$ . Then  $\mathbf{u} + l \cdot \mathbf{I}_e$  denotes the new vector generated by assigning 0 of vector  $\mathbf{u}$  to  $l$ , where  $\mathbf{I}_e$  means a unit vector corresponding to the elements in  $|D|$  with 1 in the position of  $e$ . We also denote

$$\begin{aligned} f_{\mathbf{u}}(e, l) &= f(\mathbf{u} + l \cdot \mathbf{I}_e) - f(\mathbf{u}) \\ &= f(U_1, \dots, U_{l-1}, U_l \cup \{e\}, U_{l+1}, \dots, U_k) - f(U_1, \dots, U_k), \end{aligned}$$

for  $\mathbf{u} \in (k+1)^D$ ,  $e \notin \text{supp}(\mathbf{u})$ , and  $l \in [k]$ . This is also known as a marginal gain of putting  $e \in D$  to the  $l$ -th position of  $\mathbf{u}$ . We can get that the monotonicity of  $f$  implies that  $f_{\mathbf{u}}(e, l) \geq 0$  for any  $\mathbf{u} = (U_1, \dots, U_k)$ ,  $e \notin \text{supp}(\mathbf{u})$  and  $l \in [k]$ . Moreover, the *orthant submodularity* of  $k$ -submodular is given by

$$f_{\mathbf{u}}(e, l) \geq f_{\mathbf{v}}(e, l),$$

for any  $\mathbf{u}, \mathbf{v} \in (k+1)^D$  satisfying  $\mathbf{u} \preceq \mathbf{v}$ ,  $e \notin \text{supp}(\mathbf{v})$ , and  $l \in [k]$ . The *pairwise monotonicity* of a  $k$ -submodular is known as:

$$f_{\mathbf{u}}(e, l) + f_{\mathbf{u}}(e, n) \geq 0,$$

for any element  $\mathbf{u}$  in  $(k+1)^D$ ,  $e$  not included in the support of  $\mathbf{u}$ , and any different positions  $l, n$ .

The following theorem indicates the relation among  $k$ -submodularity, orthant submodularity and pairwise monotonicity.

**Theorem 1** (Ward and Živný [14]). *A function  $f: (k+1)^D \rightarrow \mathbb{R}$  is  $k$ -submodular if and only if  $f$  is orthant submodular and pairwise monotone.*

Now we introduce a special constraint, named matroid.

**Definition 1** (Sakaue [10]). *Assuming  $D$  is a finite set and  $\mathcal{F}$  is a subset of its power set, we call the system  $(D, \mathcal{F})$  a matroid if it satisfies the following conditions:*

1.  $\emptyset \in \mathcal{F}$ ,
2. If  $A \subseteq B \in \mathcal{F}$  then  $A \in \mathcal{F}$ ,
3. If  $A, B \in \mathcal{F}$  and  $|A| < |B|$ , there must be an element  $e \in B \setminus A$  such that  $A \cup \{e\} \in \mathcal{F}$ .

Any element in  $\mathcal{F}$  is called an independent set, and  $A \in \mathcal{F}$  is named a maximal independent set if for any  $B$  satisfying  $A \subsetneq B$ , we have  $B \notin \mathcal{F}$ . In fact, the maximal independent set is also called a basis. In this paper,  $\mathcal{B}$  denotes the set of bases in  $\mathcal{F}$ ,

and every element in  $\mathcal{B}$  has the same size known as the rank of this matroid, which is denoted by  $N$  in our paper.

Given a matroid  $(D, \mathcal{F})$  and a  $k$ -submodular function  $f : (k+1)^D \rightarrow \mathbb{R}$ , the maximization  $k$ -submodular problem with a matroid constraint (MkSM) can be represented as follows.

$$\max_{\mathbf{u} \in (k+1)^D} f(\mathbf{u}) \quad \text{subject to } \text{supp}(\mathbf{u}) \in \mathcal{F}. \quad (1)$$

If the objective function is non-monotone, we denote MkSM by nMkSM. Assume that  $\bar{\mathbf{u}}$  is a feasible solution to problem (1),  $\bar{\mathbf{u}}$  is called a *maximal solution* if it has the maximal support size among all the feasible solution with the same objective value, i.e.,

$$|\text{supp}(\bar{\mathbf{u}})| \in \arg \max_{\text{supp}(\mathbf{v}) \in \mathcal{F}} \{|\text{supp}(\mathbf{v})| : f(\mathbf{v}) = f(\bar{\mathbf{u}})\}.$$

In this paper, we pay attention to the non-monotone case, and we can show that the size of its maximal solution can still reach  $N$  [10].

**Lemma 1.** *For nMkSM, the size of any maximal optimal solution is still  $N$ .*

**Proof.** Assume  $\mathbf{o}$  is an optimal solution with maximal size to which elements can no longer be added, satisfying  $|\text{supp}(\mathbf{o})| < N$ . Because  $f$  is pairwise monotone, by adding an arbitrary element  $e$  not in  $\text{supp}(\mathbf{o})$  but satisfying  $\{e\} \cup \text{supp}(\mathbf{o}) \in \mathcal{F}$  to any two different positions  $l$  and  $n$ , we have nonnegative marginal returns, that is,  $f_{\mathbf{o}}(e, l) + f_{\mathbf{o}}(e, n) \geq 0$ . Therefore, at least one of  $f_{\mathbf{o}}(e, l)$  and  $f_{\mathbf{o}}(e, n)$  is nonnegative. Suppose that  $f_{\mathbf{o}}(e, l) \geq 0$ , then the value of  $\mathbf{o} + l \cdot \mathbf{I}_e$  does not decrease the one of  $\mathbf{o}$ . This contradicts that the size of  $\mathbf{o}$  is maximal.

**Lemma 2** (Sakaue [4]). *Given a matroid  $(D, \mathcal{F})$ , assume that  $A$  is an independent set and  $B$  is a basis containing  $A$ , then for any element  $e \in D$  not in  $A$  such that  $A \cup \{e\}$  belongs to  $\mathcal{F}$ , there should be an element  $e'$  in  $B \setminus A$  satisfying that  $(B \setminus \{e'\}) \cup \{e\}$  is a basis too.*

In this paper,  $|D|$  denotes the number of elements in  $D$ ,  $p$  is the times of oracle to calculate whether a set is an independent set in this matroid, and  $q$  is the times of oracle to calculate a value of the  $k$ -submodular function.

### 3 Main Results for nMkSM

In this part, we mainly give our analysis to the nMkSM, and design two  $1/3$ -approximation algorithms for this case depending on deterministic and random techniques.

#### 3.1 A Deterministic Algorithm for nMkSM

Based on the greedy algorithm, a deterministic algorithm for nMkSM is shown in Algorithm 1. According to Lemma 1, the final output of the algorithm must be a base, so the algorithm needs to carry out  $N$ -step iteration, in which the elements in the iteration are those elements that can form an independent set with the support of the current solution (reflected in the third step of the algorithm). Finally, the greedy algorithm selects the elements and positions in the constructed subset.



**Algorithm 1** A deterministic algorithm for nMkSM

**Input:** A non-monotone  $k$ -submodular function  $f : (k+1)^D \rightarrow \mathbb{R}_+$ , a matroid  $(D, \mathcal{F})$  with bases  $\mathcal{B}$  and rank  $N \in \mathbb{Z}_+$ , and two probabilities of failure  $\varepsilon_1, \varepsilon_2$ .

**Output:** A vector  $\mathbf{t}$  with  $\text{supp}(\mathbf{t}) \in \mathcal{B}$ .

1.  $\mathbf{t} \leftarrow \mathbf{0}$  and  $n \leftarrow 1$ .
2. while  $n \leq N$
3.   Construct a set  $D^n(\mathbf{t}) = \{e \in D \setminus \text{supp}(\mathbf{t}^{n-1}) \mid \text{supp}(\mathbf{t}^{n-1}) \cup \{e\} \in \mathcal{F}\}$ .
4.   Let  $(e, l)$  be the element and position maximizing  $f_t(e, l)$  for  $e \in D^n(\mathbf{t}), l \in [k]$ .
5.    $\mathbf{t}_e \leftarrow l$ .
6.    $n \leftarrow n + 1$ .
7. end while
8. **return**  $\mathbf{t}$ .

**Theorem 2.** For nMkSM, we can obtain a  $1/3$ -approximation solution in complexity of  $O(|D|N(p+kq))$  by applying Algorithm 1.

*Proof.* In fact, the complexity is trivial and we mainly proof the performance guarantee. At beginning, let us analyze the exchange of elements between the solution output by Algorithm 1 and the optimal solution. First, taking any  $n \in [N]$ , let  $e^{(n)}, l^{(n)}$  be the element and position chosen by greedily in the  $n$ -th iteration in Algorithm 1 and let  $m^{(n)} \in [k]$  be any position other than the greedy one  $l^{(n)}$ . We say  $\mathbf{t}^{(0)} = \mathbf{0}$  and  $\mathbf{t} = \mathbf{t}^{(N)}$ , which are the output of Algorithm 1.  $\mathbf{t}^{(n)}$  represents the algorithm solution after the  $n$ -th iteration, i.e.,  $\mathbf{t}^{(n)} = \mathbf{t}^{(n-1)} + l^{(n)} \cdot \mathbf{I}_{e^{(n)}}$ . We also use  $T^{(n)}$  to denote the support of  $\mathbf{t}^{(n)}$ , i.e.,  $T^{(n)} = \text{supp}(\mathbf{t}^{(n)})$ . Let  $\mathbf{o}$  be a maximal optimal solution. We will describe the sequence  $\mathbf{o}^{(n)}$ , for  $n = 0, \dots, N$ , among them,  $\mathbf{o}^{(0)} = \mathbf{o}$ ,  $\mathbf{o}^{(N)} = \mathbf{t}$ . Here we denote  $O^{(n)} = \text{supp}(\mathbf{o}^{(n)})$ . In addition, we let  $L^n = O^{(n-1)} \setminus T^{(n-1)}$ .

Now, we explain how a series of vectors  $\mathbf{o}^{(n)}, n = (1, \dots, N)$  are constructed, which need to satisfy the following conditions:

$$\mathbf{t}^{(n)} \begin{cases} \prec \mathbf{o}^{(n)}, & \text{if } n = 1, 2, \dots, N-1, \\ = \mathbf{o}^{(n)} & \text{if } n = N. \end{cases}$$

and  $O^{(n)}$  is an element of  $\mathcal{B}$  for  $n = 1, 2, \dots, N$ . We use the following ways to exchange elements for constructing the sequence:

$$\mathbf{o}^{(n)} = \begin{cases} e^{(n)}, & \text{if } e^{(n)} \in L^n, \\ \text{any element in } L^n, & \text{otherwise.} \end{cases}$$

$$\begin{aligned} \mathbf{o}^{(n-1/2)} &= \mathbf{o}^{(n-1)} - \mathbf{o}^{(n-1)}(o^{(n)}) \cdot \mathbf{I}_{o^{(n)}}, \\ \mathbf{o}^{(n)} &= \mathbf{o}^{(n-1/2)} + l^{(n)} \cdot \mathbf{I}_{e^{(n)}}. \end{aligned}$$

Thus, since  $e^{(n)}$  and  $l^{(n)}$  are chosen greedily in the  $n$ -th step of Algorithm 1, we get:

$$f_{\mathbf{t}^{(n-1)}}(e^{(n)}, l^{(n)}) \geq f_{\mathbf{t}^{(n-1)}}(\mathbf{o}^{(n)}, \mathbf{o}^{(n-1)}(o^{(n)})), \quad (2)$$

$$f_{\mathbf{t}^{(n-1)}}(e^{(n)}, l^{(n)}) \geq f_{\mathbf{t}^{(n-1)}}(\mathbf{o}^{(n)}, m^{(n)}). \quad (3)$$

Since  $\mathbf{t}^{(n-1)} \preceq \mathbf{o}^{(n-1/2)}$ , we have:

$$f_{\mathbf{t}^{(n-1)}}(\mathbf{o}^{(n)}, \mathbf{o}^{(n-1)}(\mathbf{o}^{(n)})) \geq f_{\mathbf{o}^{(n-1/2)}}(\mathbf{o}^{(n)}, \mathbf{o}^{(n-1)}(\mathbf{o}^{(n)})), \quad (4)$$

$$f_{\mathbf{t}^{(n-1)}}(e^{(n)}, m^{(n)}) \geq f_{\mathbf{o}^{(n-1/2)}}(e^{(n)}, m^{(n)}). \quad (5)$$

And by the pairwise monotonicity of  $f$  and  $m^{(n)} \neq l^{(n)}$ , we have

$$f_{\mathbf{o}^{(n-1/2)}}(e^{(n)}, l^{(n)}) + f_{\mathbf{o}^{(n-1/2)}}(e^{(n)}, m^{(n)}) \geq 0. \quad (6)$$

Next, we deduce the proof as follows.

$$\begin{aligned} & f(\mathbf{t}^{(n)}) - f(\mathbf{t}^{(n-1)}) \\ &= f_{\mathbf{t}^{(n-1)}}(e^{(n)}, l^{(n)}) \\ &\stackrel{(2)}{\geq} f_{\mathbf{t}^{(n-1)}}(\mathbf{o}^{(n)}, \mathbf{o}^{(n-1)}(\mathbf{o}^{(n)})) \\ &\stackrel{(4)}{\geq} f_{\mathbf{o}^{(n-1/2)}}(\mathbf{o}^{(n)}, \mathbf{o}^{(n-1)}(\mathbf{o}^{(n)})) \\ &\stackrel{(6)}{\geq} f_{\mathbf{o}^{(n-1/2)}}(\mathbf{o}^{(n)}, \mathbf{o}^{(n-1)}(\mathbf{o}^{(n)})) - f_{\mathbf{o}^{(n-1/2)}}(e^{(n)}, l^{(n)}) - f_{\mathbf{o}^{(n-1/2)}}(e^{(n)}, m^{(n)}) \\ &\stackrel{(5)}{\geq} f_{\mathbf{o}^{(n-1/2)}}(\mathbf{o}^{(n)}, \mathbf{o}^{(n-1)}(\mathbf{o}^{(n)})) - f_{\mathbf{o}^{(n-1/2)}}(e^{(n)}, l^{(n)}) - f_{\mathbf{t}^{(n-1)}}(e^{(n)}, m^{(n)}) \\ &\stackrel{(3)}{\geq} f_{\mathbf{o}^{(n-1/2)}}(\mathbf{o}^{(n)}, \mathbf{o}^{(n-1)}(\mathbf{o}^{(n)})) - f_{\mathbf{o}^{(n-1/2)}}(e^{(n)}, l^{(n)}) - f_{\mathbf{t}^{(n-1)}}(e^{(n)}, l^{(n)}) \\ &= f(\mathbf{o}^{(n-1)}) - f(\mathbf{o}^{(n-1/2)}) - f_{\mathbf{o}^{(n-1/2)}}(e^{(n)}, l^{(n)}) - f_{\mathbf{t}^{(n-1)}}(e^{(n)}, l^{(n)}). \end{aligned} \quad (7)$$

By transposing the two sides of the inequality in (7), we have,

$$\begin{aligned} 2f_{\mathbf{t}^{(n-1)}}(e^{(n)}, l^{(n)}) &\geq f(\mathbf{o}^{(n-1)}) - f(\mathbf{o}^{(n-1/2)}) - f_{\mathbf{o}^{(n-1/2)}}(e^{(n)}, l^{(n)}) \\ &= f(\mathbf{o}^{(n-1)}) - f(\mathbf{o}^{(n-1/2)}) - f(\mathbf{o}^{(n)}) + f(\mathbf{o}^{(n-1/2)}) \\ &= f(\mathbf{o}^{(n-1)}) - f(\mathbf{o}^{(n)}). \end{aligned}$$

Thus,

$$2(f(\mathbf{t}^{(n)}) - f(\mathbf{t}^{(n-1)})) \geq f(\mathbf{o}^{(n-1)}) - f(\mathbf{o}^{(n)}). \quad (8)$$

By summing over both sides of (8), we get the following results:

$$\begin{aligned} f(\mathbf{o}) - f(\mathbf{t}) &= \sum_{n=1}^N (f(\mathbf{o}^{(n-1)}) - f(\mathbf{o}^{(n)})) \\ &\leq 2 \sum_{n=1}^N (f(\mathbf{t}^{(n)}) - f(\mathbf{t}^{(n-1)})) \\ &= 2f(\mathbf{t}). \end{aligned}$$

Finally, we draw the following conclusion:

$$f(\mathbf{t}) \geq 1/3f(\mathbf{o}).$$

In the following subsection, we will design a random algorithm to improve the complexity of the deterministic algorithm from  $O(N|D|(p+kq))$  to  $O(|D|(p \log \frac{N}{\varepsilon_1} + kq \log \frac{N}{\varepsilon_2}) \log N)$  with probability at least  $1 - \varepsilon$ , where  $\varepsilon = \max\{\varepsilon_1, \varepsilon_2\}$ .

### 3.2 A Random Algorithm for nMkSM

In order to reduce the complexity of the deterministic algorithm for the non-monotone case, we adopt the uniform random sampling, which is shown as Algorithm 2.

There are two major difference between the random algorithm and the deterministic algorithm. Firstly, we add random  $Q^n$  in Step 3 to reduce the number of elements for constructing the independent sets, that is, the element is selected from  $Q^n \setminus \text{supp}(\mathbf{t}^{n-1})$  instead of  $D \setminus \text{supp}(\mathbf{t}^{n-1})$  when constructing the independent set  $D^n(\mathbf{t})$ , where  $D^n(\mathbf{t}) = \{e \in Q^n \setminus \text{supp}(\mathbf{t}^{n-1}) \mid \text{supp}(\mathbf{t}^{n-1}) \cup \{e\} \in \mathcal{F}\}$ . Secondly, we use random set  $R^n$  in Step 5 to reduce the size of  $D^n(\mathbf{t})$ . Under the condition that the constructed independent set is non-empty, some elements in the independent set are randomly selected to form the random set of  $R^n$ , instead of  $D^n(\mathbf{t})$  when it makes greedy selection.

---

#### Algorithm 2 A random algorithm for nMkSM

---

**Input:** A non-monotone  $k$ -submodular function  $f : (k+1)^D \rightarrow \mathbb{R}_+$ , a matroid  $(D, \mathcal{F})$  with bases  $\mathcal{B}$  and rank  $N \in \mathbb{Z}_+$ , and two failed probabilities  $\varepsilon_1, \varepsilon_2$ .

**Output:** A vector  $\mathbf{t}$  with  $\text{supp}(\mathbf{t}) \in \mathcal{B}$ .

1.  $\mathbf{t} \leftarrow \mathbf{0}$  and  $n \leftarrow 1$ .
  2. while  $n \leq N$
  3.  $Q^n \leftarrow$  a random subset uniformly picked from  $D$ , whose size is  $\min\{\frac{|D|-n+1}{N-n+1} \log(\frac{N}{\varepsilon_1}), |D|\}$ .
  4. Construct  $D^n(\mathbf{t})$  using the independence oracle.
  5.  $R^n \leftarrow$  a random subset evenly selected from  $D^n(\mathbf{t})$  with size of  $\min\{\frac{|Q^n|-n+1}{N-n+1} \log(\frac{N}{\varepsilon_2}), |D^n(\mathbf{t})|\}$ .
  6.  $(e, l) \leftarrow \arg \max_{e \in R^n, l \in [k]} f_t(e, l)$ .
  7.  $\mathbf{t}(e) \leftarrow l$ .
  8.  $n \leftarrow n + 1$ .
  9. end while
  10. **return**  $\mathbf{t}$ .
- 

In Algorithm 2, we first append  $\mathbf{0}$  to  $\mathbf{t}$  and then go through a loop. Each loop takes the following 6 steps from step 3 to step 8, for a total of  $N$  loops, and finally outputs a vector  $\mathbf{t}$  that belongs to  $\mathcal{B}$ . Below we specifically analyze each cycle of six steps. The first step (the step 3 in Algorithm 2) is to randomly select some elements to form  $Q^n$  from the set  $D$ , but the size of  $Q^n$  is limited. The second step (the step 4 in Algorithm 2) is to form the qualified  $e \in Q^n$  into an independent set  $D^n(\mathbf{t})$ . The third step (the step 5 in Algorithm 2) is to randomly select the elements to form the  $R^n$  in  $D^n(\mathbf{t})$ , which greatly reduces the number of selected elements. The fourth step (the step 6 in Algorithm 2) is to greedily select the elements and positions in  $R^n$  until they are found which can make the  $k$ -submodular function produce the maximum gain. The selected element is written as  $e$  and the selected position is recorded as  $l$ . Then we can obtain the following result.

**Lemma 3.** We can get  $D^n(\mathbf{t}) \neq \emptyset$  with the probability at least  $1 - \varepsilon_1$ . Likewise, with probability at least  $1 - \varepsilon_2$ , we can also obtain  $R^n \cap L^n \neq \emptyset$  for every  $n \in [N]$ .

**Proof.** First, for  $n \in [N]$ , if  $|Q^n| = |D|$ , there is obviously  $D^n(\mathbf{t}) \neq \emptyset$ . That is,  $\Pr[D^n(\mathbf{t}) = \emptyset] = 0$ . Otherwise we have

$$\begin{aligned} \Pr[D^n(\mathbf{t}) = \emptyset] &= (\Pr[\{e\} \cup \text{supp}(\mathbf{t}) \notin \mathcal{F}])^{|Q^n|} \\ &= (1 - \Pr[\{e\} \cup \text{supp}(\mathbf{t}) \in \mathcal{F}])^{|Q^n|} \\ &\leq \left(1 - \frac{N - (n - 1)}{|Q^n| - (n - 1)}\right)^{|Q^n|} \\ &\leq \left(1 - \frac{N - (n - 1)}{|D| - (n - 1)}\right)^{|Q^n|} \\ &\leq e^{-\frac{N-n+1}{|D|-n+1}|Q^n|} \\ &= e^{-\frac{N-n+1}{|D|-n+1} \frac{|D|-n+1}{N-n+1} \log \frac{N}{\varepsilon_1}} \\ &= \frac{\varepsilon_1}{N}. \end{aligned}$$

In the same way, for  $n \in [N]$ , if  $|R^n| = |D^n(\mathbf{t})|$ , obviously there is  $R^n \cap L^n \neq \emptyset$ , then  $\Pr[R^n \cap L^n = \emptyset] = 0$ . Otherwise we have

$$\begin{aligned} \Pr[R^n \cap L^n = \emptyset] &= \left(1 - \frac{|L^n|}{|D^n(\mathbf{t})|}\right)^{|R^n|} \\ &= \left(1 - \frac{N - (n - 1)}{|D^n(\mathbf{t})|}\right)^{|R^n|} \\ &\leq \left(1 - \frac{N - (n - 1)}{|Q^n| - (n - 1)}\right)^{|R^n|} \\ &\leq e^{-\frac{N-n+1}{|Q^n|-n+1}|R^n|} \\ &= e^{-\frac{N-n+1}{|Q^n|-n+1} \frac{|Q^n|-n+1}{N-n+1} \log \frac{N}{\varepsilon_2}} \\ &= \frac{\varepsilon_2}{N}. \end{aligned}$$

Taking any  $n \in [N]$ , we will construct a sequence of the optimal solution  $\mathbf{o}^{(0)} = \mathbf{o}, \mathbf{o}^{(1)}, \dots, \mathbf{o}^{(N)} = \mathbf{t}$ . Denote the symbols  $e^{(n)}, l^{(n)}, \mathbf{t}^{(n)}, T^{(n)}, O^{(n)}, L^n$  as the same meanings as those in deterministic case (see the proof of Theorem 2). If  $D^n(\mathbf{t})$  or  $R^n \cap L^n$  is empty, we consider the algorithm failed. Suppose  $D^n(\mathbf{t})$  and  $R^n \cap L^n$  is non-empty, we have the following definitions.

$$\mathbf{o}^{(n)} = \begin{cases} e^{(n)}, & \text{if } e^{(n)} \in R^n \cap L^n, \\ \text{any element in } R^n \cap L^n, & \text{otherwise.} \end{cases}$$

Finally, we let

$$\begin{aligned} \mathbf{o}^{(n-1/2)} &= \mathbf{o}^{(n-1)} - \mathbf{o}^{(n-1)}(\mathbf{o}^{(n)}) \cdot \mathbf{I}_{\mathbf{o}^{(n)}}, \\ \mathbf{o}^{(n)} &= \mathbf{o}^{(n-1/2)} + l^{(n)} \cdot \mathbf{I}_{e^{(n)}}. \end{aligned}$$

Based on the construction of the optimal solution above and just like the deterministic Algorithm 1, if the algorithm does not fail, we have  $\mathbf{o}^{(N)} = \mathbf{t}^{(N)}$  and can get

$$2(f(\mathbf{t}^{(n)}) - f(\mathbf{t}^{(n-1)})) \geq f(\mathbf{o}^{(n-1)}) - f(\mathbf{o}^{(n)}), \forall n = 1, 2, \dots, N.$$

Thus,

$$3f(\mathbf{t}) \geq f(\mathbf{o}).$$

In the following part, we will give the proof of Theorem 3 based on Lemma 3, which shows that Algorithm 2 does not fail with high probability.

**Theorem 3.** *For nMkSM, we can obtain a  $1/3$ -approximation solution from Algorithm 2 in complexity of  $O(|D|(p \log \frac{N}{\varepsilon_1} + kq \log \frac{N}{\varepsilon_2}) \log N)$  with the probability at least  $1 - \varepsilon$ , where  $\varepsilon = \max\{\varepsilon_1, \varepsilon_2\}$ .*

**Proof.** The number of times Algorithm 2 is at most

$$\begin{aligned} p \sum_{n=1}^N |Q^n| + kq \sum_{n=1}^N |R^n| &= p \log \frac{N}{\varepsilon_1} \sum_{n=1}^N \frac{|D| - n + 1}{N - n + 1} + kq \log \frac{N}{\varepsilon_2} \sum_{n=1}^N \frac{|Q^n| - n + 1}{N - n + 1} \\ &= p \log \frac{N}{\varepsilon_1} \sum_{n=1}^N \frac{|D| - N + n}{n} + kq \log \frac{N}{\varepsilon_2} \sum_{n=1}^N \frac{|Q^n| - N + n}{n} \\ &\leq p \log \frac{N}{\varepsilon_1} \sum_{n=1}^N \frac{|D|}{n} + kq \log \frac{N}{\varepsilon_2} \sum_{n=1}^N \frac{|D|}{n} \\ &\leq |D| (p \log \frac{N}{\varepsilon_1} \log N + kq \log \frac{N}{\varepsilon_2} \log N). \end{aligned}$$

## References

1. Fujishige, S., Iwata, S.: Bisubmodular function minimization. *SIAM J. Discret. Math.* **19**, 1065–1073 (2006)
2. Huber, A., Kolmogorov, V.: Towards minimizing  $k$ -submodular functions. In: Proceedings of ISCO, pp. 451–462 (2012)
3. Iwata, S., Tanigawa, S., Yoshida, Y.: Improved approximation algorithms for  $k$ -submodular function maximization. In: Proceedings of SODA, pp. 404–413 (2016)
4. Korte, B., Vygen, J.: *Combinatorial Optimization: Theory and Algorithms*, 5th edn. Springer (2011)
5. Ohsaka, N., Yoshida, Y.: Monotone  $k$ -submodular function maximization with size constraints. In: Proceedings of NIPS, pp. 694–702 (2015)
6. Oshima, H.: Improved randomized algorithm for  $k$ -submodular function maximization. *SIAM J. Discret. Math.* **35**, 1–22 (2021)
7. Pham, C.V., Vu, Q.C., Ha, D.K.T., Nguyen, T.T.: Streaming algorithms for budgeted  $k$ -submodular maximization problem. In: Proceeding of CSoNet, pp. 27–38 (2021)
8. Qian, C., Shi, J., Tang, K., Zhou, Z.: Constrained monotone  $k$ -submodular function maximization using multiobjective evolutionary algorithms with theoretical guarantee. *IEEE Trans. Evol. Comput.* **22**, 595–608 (2018)

9. Rafiey, A., Yoshida, Y.: Fast and private submodular and  $k$ -submodular functions maximization with matroid constraints. In: Proceeding of ICML, pp. 7887–7897 (2020)
10. Sakaue, S.: On maximizing a monotone  $k$ -submodular function subject to a matroid constraint. *Discret. Optim.* **23**, 105–113 (2017)
11. Tang, Z., Wang, C., Chan, H.: Monotone  $k$ -submodular secretary problems: cardinality and knapsack constraints. *Theoret. Comput. Sci.* **921**, 86–99 (2022)
12. Tang, Z., Wang, C., Chan, H.: On maximizing a monotone  $k$ -submodular function under a knapsack constraint. *Oper. Res. Lett.* **50**, 28–31 (2022)
13. Wang, B., Zhou, H.: Multilinear extension of  $k$ -submodular functions. [arXiv:2107.07103](https://arxiv.org/abs/2107.07103) (2021)
14. Ward, J., Živný, S.: Maximizing  $k$ -submodular functions and beyond. *ACM T. Algorithms*, 1–26, 47 (2016)



# Maximizing Approximately Non- $k$ -Submodular Monotone Set Function with Matroid Constraint

YanJun Jiang<sup>1</sup>, Yijing Wang<sup>2</sup>, Ruiqi Yang<sup>3,4(✉)</sup>, and Weina Ye<sup>3</sup>

<sup>1</sup> School of Mathematics and Statistics Science, Ludong University, Yantai 264025, People's Republic of China

<sup>2</sup> Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, People's Republic of China

<sup>3</sup> Beijing Institute for Scientific and Engineering Computing, Beijing University of Technology, Beijing 100124, People's Republic of China

<sup>4</sup> School of Mathematical Sciences, University of Chinese Academy Sciences, Beijing 100049, People's Republic of China  
yangruiqi@ucas.ac.cn

**Abstract.** Utilizing approximation algorithm, there has been a large quantity of work on optimization for submodular functions since the 1970s. As a variant,  $k$ -submodular function appears in many fields to match with the developing background, in which the outputted sets changes from one to  $k$ . Because of the application of submodularity, some concepts and parameters describing the closeness to submodular function are generated, for example approximately submodular set function and diminishing-return (DR) ratio. In our discussion, the  $k$ -dimension set function with matroid constraint we will maximize may not have access to the submodularity. However it is an approximately non- $k$ -submodular set function which contains DR ratio. By the greedy technique, we obtain the approximation algorithms. When the value of the DR ratio is set one, some known results are covered.

**Keywords:**  $k$ -submodular set function · Matroid · Greedy · Approximation algorithm

## 1 Introduction

A non-negative set function  $h : 2^G \rightarrow \mathbb{R}$  defined on subsets of a ground set  $G$  with size  $n$  is called submodular if it possesses the following properties. One is that  $\forall S_1 \subseteq S_2 \subseteq G$ ,  $h(S_1) + h(S_2) \geq h(S_1 \cup S_2) + h(S_1 \cap S_2)$ ,  $\forall S_1, S_2 \subseteq G$ . There is also an intuitive diminishing return property for submodularity  $h(u|S_2) \leq h(u|S_1)$ ,  $\forall S_1 \subseteq S_2 \subseteq G$ ,  $u \in G \setminus S_2$ , where  $h(u|S) := h(S \cup \{u\}) - h(S)$  indicates the marginal gain of  $h$  with respect to any element  $u$  and any subset  $S \subseteq G$ . Submodular functions appear in classic optimization problems such as the maximization of facility location, weighted cut, and set cover etc. In recent modern subjects such as machine learning and data mining, there are also some problems concerning the objective function with diminishing return property for example monitor placement [1], dictionary selection [2], sensor placement [3] and social networks [4].

Instead of selecting one set, several disjoint sets are required in recent practical applications. Therefore,  $k$ -submodular set function is a natural generalization of 1-dimension submodular set function. The special cases  $k = 1$  and  $k = 2$  are submodular and bisubmodular functions respectively. There are some obvious examples, such as influence maximization [5] and sensor placement [5]. According to the actual application background, the maximization problems are discussed for  $k$ -submodular set function with various constraints. For the maximization of nonnegative monotone  $k$ -submodular function, Ohsaka and Yoshida [5] provide constant-factor algorithms, in which there are two kinds of size constraints. For matroid constraint, Sakauc [6] proposes a  $\frac{1}{2}$ -approximation algorithm with greedy technique. Tang *et al.* [7] obtain a deterministic  $\frac{1}{2} - \frac{1}{2e}$  under a knapsack constraint.

As submodularity for set function is widely used in practical settings, there are many related parameters describing submodular functions, or concepts introducing the proximity between set functions and submodular set functions, for example weak submodular ratio, diminishing return ratio [8,9] and approximately submodular function [10,11]. Therefore, the corresponding description can be extended to  $k$ -submodular set function as well.

In many applying situations, there is no access to the submodular function to be optimized. However, it prefers to some erroneous or noisy version. Therefore, approximately submodular set function and submodular optimization under noise [12,13] are discussed later. In these cases, the optimization for a non-negative monotone set function can be achieved, relating some non-negative monotone submodular classic set function. When the number of the selected subsets varies from one to  $k$  ( $k \geq 2$ ), maximizing approximately submodular functions is studied, in which various conditions are given. Horel and Singer [13] have shown an approximation algorithm for approximately  $k$ -submodular functions. In the discussion of  $k$ -dimension set functions, Zheng *et al.* [10] give two kinds of definitions. One is  $\epsilon$ -approximately  $k$ -submodular function ( $\epsilon$ -AS), the other is  $\epsilon$ -approximately diminishing return ( $\epsilon$ -ADR). Utilizing greedy method, they acquire approximation algorithms under the cardinality settings of total size and individual size constraints. Recently, Matsuoka and Ohsaka [14] consider  $\epsilon$ -approximately  $k$ -submodular function ( $\epsilon$ -AS) with matroid constraint.

In this work, we consider the approximation of maximization for non- $k$ -submodular monotone set function with noise. In the setting, a  $k$ -dimension set function with matroid constraint is to be investigated, which is approximately non- $k$ -submodular. The greedy technique is adapted to deal with the maximization of set function with matroid constraint. After the considerable analysis, we obtain the approximation ratios of two kinds of situations for the set function we consider. One is the maximization of  $\epsilon$ -approximately  $\alpha$ -weakly  $k$ -submodular ( $\epsilon$ -A- $\alpha$ -WS) with matroid constraint, the approximation ratio is  $\frac{\alpha(1-\epsilon)^2}{(1+\epsilon)[(\alpha+1)(1-\epsilon)+2B\epsilon]}$ . The other is the maximization of  $\epsilon$ -approximately  $\alpha$ -weakly diminishing return ( $\epsilon$ -A- $\alpha$ -WDR), the approximation ratio is  $\frac{(1-\epsilon)\alpha}{1+\epsilon+\alpha-\epsilon\alpha}$ . When the parameter  $\alpha$  for weakly  $k$ -submodular set function is set with 1, our conclusions generalize some known results [14] correspondingly.

The outline of this article is as follows. Preliminaries are introduced in Sect. 2. In Sect. 3, there are the greedy algorithms and the analysis for the  $\epsilon$ -A- $\alpha$ -WS and  $\epsilon$ -A- $\alpha$ -WDR, resulting the approximation ratios. The concluding remarks are in Sect. 4.



## 2 Preliminaries

The cardinality of ground set  $G$  is denoted with non-negative integer  $n$ , and  $[n]$  is identified with  $G$  for convenience. For any integer  $k \in \mathbb{N}$ ,  $[k]$  denotes set  $\{1, 2, \dots, k\}$ . We set  $(k+1)^G := \{\mathbf{s} = (S_1, S_2, \dots, S_k) \mid S_l \subseteq G, \forall l \in \{1, 2, \dots, k\}, S_i \cap S_j = \emptyset, \forall i \neq j\}$ . A partial order  $\preceq$  is defined on  $(k+1)^G$ , that is, for  $\mathbf{s} = (S_1, S_2, \dots, S_k), \mathbf{t} = (T_1, T_2, \dots, T_k) \in (k+1)^G$ , if  $\mathbf{s} \sqcup \mathbf{t} = \mathbf{t}$  (i.e.,  $S_l \subseteq T_l, \forall l \in [k]$ ) holds, then the partial order  $\mathbf{s} \preceq \mathbf{t}$  is satisfied. Similar to the concept of marginal benefit for one-dimension set function, there is the concept of marginal benefit for multi-dimension set function as well. For set function  $h(\mathbf{s})$  defined on  $(k+1)^G$ ,  $\Delta_{u,i}h(\mathbf{s}) = h(S_1, S_2, \dots, S_{i-1}, S_i \cup \{u\}, S_{i+1}, \dots, S_k) - f(S_1, S_2, \dots, S_k)$ , for  $\mathbf{s} \in (k+1)^G, u \notin \bigcup_{t \in [k]} S_t$  and  $i \in [k]$ . This illustrates the marginal benefit for multi-dimension set function  $h(\mathbf{s})$  when element  $u$  is added into the  $i$ -th set of  $\mathbf{s}$ . In this paper, we make a slight abuse of notion used by [5]. We associate each  $\mathbf{s} = (S_1, S_2, \dots, S_k) \in (k+1)^G$  with  $\mathbf{s} \in \{0, 1, \dots, k\}^G$  by  $S_i = \{u \in G : \mathbf{s}(u) = i\}$  for  $i \in [k]$ . For  $\mathbf{s} \in (k+1)^G$ , we define  $\text{supp}(\mathbf{s}) = \{u \in G : \mathbf{s}(u) \neq 0\}$ . Similarly, for  $\mathbf{s} \in (k+1)^G, i \in [k]$ ,  $\text{supp}_i(\mathbf{s}) = \{u \in G : \mathbf{s}(u) = i\}$  is defined. Let  $\mathbf{0}$  be a vector of  $k$  empty sets, i.e.,  $\mathbf{0} = (S_1 = \emptyset, \dots, S_k = \emptyset)$ .

**Definition 1 (*k*-submodular set function).** *Set function  $h : (k+1)^G \rightarrow \mathbb{R}$  is called  $k$ -submodular if and only if, for any  $\mathbf{s} = (S_1, \dots, S_k), \mathbf{t} = (T_1, \dots, T_k)$ , we have  $h(\mathbf{s}) + h(\mathbf{t}) \geq h(\mathbf{s} \sqcup \mathbf{t}) + h(\mathbf{s} \sqcap \mathbf{t})$ , where  $\mathbf{s} \sqcap \mathbf{t} := (S_1 \cap T_1, \dots, S_k \cap T_k), \mathbf{s} \sqcup \mathbf{t} := (S_1 \cup T_1 \setminus (\bigcup_{i \neq 1} S_i \cup T_i), \dots, S_k \cup T_k \setminus (\bigcup_{i \neq k} S_i \cup T_i))$ . When  $k = 1$ ,  $k$ -submodularity coincides with submodularity for one-dimension set function.*

**Definition 2 (monotone).** *A  $k$ -submodular set function  $h$  is monotone if and only if, for any  $\mathbf{s}, \mathbf{t} \in (k+1)^G$  such that if  $\mathbf{s} \preceq \mathbf{t}$ ,  $h(\mathbf{s}) \leq h(\mathbf{t})$  holds.*

Das and Kempe [2] introduce submodularity ratio firstly, which characterizes the closeness with the submodularity for a set function. Correspondingly, there is the definition of weakly submodular for set functions [8]. A nonnegative monotone set function  $h : 2^G \rightarrow \mathbb{R}$  is called a  $\alpha$ -weakly submodular set function if  $h$  satisfies that  $\sum_{u \in S_2} h(u|S_1) \geq \alpha h(S_2 | S_1)$ , for all disjoint  $S_1, S_2 \subseteq G$ . For one-dimension set function, there several equivalent definitions for submodularity. Therefore, there are equivalent form for  $\alpha$ -weakly submodular set function [9], which is  $h(u|S_1) \geq \alpha h(u|S_2)$ , for any  $S_1 \subseteq S_2 \subseteq G$  and  $u \notin S_2$ ,  $\alpha$  is the maximum value in  $[0, 1]$  and is called diminishing-return (DR) ratio here. Thus, a generalized concept of  $\alpha$ -weakly  $k$ -submodular set function can be yielded obviously.

**Definition 3 ( $\alpha$ -weakly  $k$ -submodular set function).** *A nonnegative monotone  $k$ -dimension set function  $h : (k+1)^G \rightarrow \mathbb{R}$  is called  $\alpha$ -weakly  $k$ -submodular if there exists the largest scalar  $\alpha \in [0, 1]$ , such that for  $\mathbf{s}, \mathbf{t} \in (k+1)^G$  satisfying  $\mathbf{s} \preceq \mathbf{t}, u \notin \text{supp}(\mathbf{t}), \forall i \in [k]$ , the following holds*

$$h((u, i)|\mathbf{s}) \geq \alpha h((u, i)|\mathbf{t}),$$

where  $\alpha$  is called the diminishing-return (DR) ratio.

In the context of the actual problem such as the influence maximization or information coverage problem, the query for submodular set function  $h$  may be not easy, which needs at least exponential number of computations [4]. In the absence of particular strictness, a noisy version for submodular function  $f$  can be characterized [11] as this. For  $\epsilon > 0$ ,  $H : 2^G \rightarrow \mathbb{R}^+$  is an  $\epsilon$ -approximately set submodular function if there is exists a submodular set function  $h : 2^G \rightarrow \mathbb{R}^+$  satisfying  $(1 - \epsilon)h(S) \leq H(S) \leq (1 + \epsilon)h(S)$  for all  $S \subseteq G$  [12]. When the dimension of the domain for the set function varies from one to  $k$ , the corresponding noisy version of submodular set functions appears, which is an  $\epsilon$ -approximately  $k$ -submodular set function [10, 11]. For  $k$ -submodular set function, Zheng *et al.* [10] give not only the discussion of  $\epsilon$ -approximately  $k$ -submodular ( $\epsilon$ -AS) but also  $\epsilon$ -approximating diminishing return ( $\epsilon$ -ADR).

Based on the definition of  $\epsilon$ -approximately  $k$ -submodular [10] and  $\alpha$ -weakly  $k$ -submodular set function, we give the definition of  $\epsilon$ -approximately  $\alpha$ -weakly  $k$ -submodular set function ( $\epsilon$ -A- $\alpha$ -WS). According to  $\epsilon$ -approximately diminishing return [10] and  $\alpha$ -weakly  $k$ -submodular set function,  $\epsilon$ -approximately  $\alpha$ -weakly diminishing return ( $\epsilon$ -A- $\alpha$ -WDR) is also proposed.

**Definition 4 ( $\epsilon$ -approximately  $\alpha$ -weakly  $k$ -submodular ( $\epsilon$ -A- $\alpha$ -WS)).** A set function  $H(s) : (k + 1)^G \rightarrow \mathbb{R}^+$  is defined  $\epsilon$ -approximately  $\alpha$ -weakly  $k$ -submodular for some small  $\epsilon > 0$ , if and only if there exists a monotone  $\alpha$ -weakly  $k$ -submodular function  $h$  such that for any  $s \in (k + 1)^G$ , satisfying

$$(1 - \epsilon)h(s) \leq H(s) \leq (1 + \epsilon)h(s).$$

**Definition 5 ( $\epsilon$ -approximately  $\alpha$ -weakly diminishing return ( $\epsilon$ -A- $\alpha$ -WDR)).** A set function  $H(s) : (k + 1)^G \rightarrow \mathbb{R}^+$  is defined  $\epsilon$ -approximately  $\alpha$ -weakly diminishing return for some small  $\epsilon > 0$ , if and only if there exists a monotone  $\alpha$ -weakly  $k$ -submodular function  $h$  such that for any  $s \in (k + 1)^G$ ,  $u \notin \cup_{l \in [k]} S_l$ ,  $u \in G$  and  $i \in [k]$ , satisfying

$$(1 - \epsilon) \Delta_{u,i} h(s) \leq \Delta_{u,i} H(s) \leq (1 + \epsilon) \Delta_{u,i} h(s).$$

**Definition 6 (matroid).** Let  $\mathcal{M} = (G, \mathcal{F})$  be a set system, in which  $G$  is a ground set with cardinality  $n$  and  $\mathcal{F} \subseteq 2^G$  is a family of subsets. If  $\mathcal{F} \neq \emptyset$ , and the subsets in it satisfy the following two conditions: (1)  $\emptyset \in \mathcal{F}$ ; and (2) for any subset  $S_1 \subseteq S_2 \subseteq G$ , if  $S_2 \in \mathcal{F}$ , then  $S_1 \in \mathcal{F}$ , then the set system  $\mathcal{M} = (G, \mathcal{F})$  is called an independent system. For an independent system  $\mathcal{M} = (G, \mathcal{F})$ ,  $\forall S_1, S_2 \in \mathcal{F}$ , if  $|S_1| < |S_2|$ , there is an element  $u \in S_2 \setminus S_1$  such that  $S_1 \cup \{u\} \in \mathcal{F}$ , then it is a matroid. The set family of maximal elements of  $\mathcal{F}$  is denoted with  $\mathcal{B}$ , which is called the base family. From the exchange property, every base in  $\mathcal{B}$  is of the same size whose cardinality is called the rank denoted with  $B$  in a matroid.

For the maximization of set function, there often exists constraints no matter the function is submodular or non-submodular. The usual constraints are cardinality constraint, knapsack constraint and matroid constraint. For  $k$ -dimension set function, there are two kinds of cardinality constraints. One is total size constraint, the other is individual constraint. A matroid constraint is a generalization of the total size constraint. A solution  $\mathbf{x} \in (k + 1)^G$  is a feasible solution if  $\text{supp}(\mathbf{x}) \in \mathcal{F}$  under the matroid constraint with a matroid  $\mathcal{M} = (G, \mathcal{F})$ .

### 3 Greedy Algorithm and Analysis

#### 3.1 Greedy Algorithm

---

**Algorithm 1** Greedy algorithm for  $\epsilon$ -A- $\alpha$ -WS with matroid constraint

---

**Input:** a matroid  $(G, \mathcal{F})$ ,  $\mathcal{B}$  is a base family,  $B \in \mathbb{Z}^+$  is the rank of the matroid and a monotone  $\epsilon$ -approximately  $\alpha$ -weakly  $k$ -submodular  $H : (k+1)^G \rightarrow \mathbb{R}^+$ .

**Output:** a vector  $s^{(B)}$  with  $|\text{supp}(s^{(B)})| = B$ .

1.  $s^{(0)} \leftarrow \mathbf{0}$
  2. for  $j = 1, 2, \dots, B$  do
  3.  $(u^{(j)}, i^{(j)}) \leftarrow \arg\max_{(u,i) \in ([n] \setminus \text{supp}(s^{(j-1)})) \times [k]} \Delta_{(u,i)} H(x)$
  4.  $s^{(j)} \leftarrow (u^{(j)}, i^{(j)}) + s^{(j-1)}$
  5. return  $s^{(B)}$
- 

---

**Algorithm 2** Greedy algorithm for  $\epsilon$ -A- $\alpha$ -WDR with matroid constraint

---

**Input:** a matroid  $(G, \mathcal{F})$ ,  $\mathcal{B}$  is a base family,  $B \in \mathbb{Z}^+$  is the rank of the matroid and a monotone  $\epsilon$ -approximately  $\alpha$ -weakly diminishing return  $H : (k+1)^G \rightarrow \mathbb{R}^+$ .

**Output:** a vector  $s^{(B)}$  with  $|\text{supp}(s^{(B)})| = B$ .

1.  $s^{(0)} \leftarrow \mathbf{0}$
  2. for  $j = 1, 2, \dots, B$  do
  3.  $(u^{(j)}, i^{(j)}) \leftarrow \arg\max_{(u,i) \in ([n] \setminus \text{supp}(s^{(j-1)})) \times [k]} \Delta_{(u,i)} H(x)$
  4.  $s^{(j)} \leftarrow (u^{(j)}, i^{(j)}) + s^{(j-1)}$
  5. return  $s^{(B)}$
- 

In the analysis of the algorithm, we use Lemma 1 in [6], in which  $\mathcal{B}$  is a base family of a matroid  $(G, \mathcal{F})$  which is mentioned in Algorithm 1 and Algorithm 2.

**Lemma 1.** *Let  $S_1 \in \mathcal{F}, S_2 \in \mathcal{B}$  with  $S_1 \subsetneq S_2$ , and  $u \notin S_1$  with  $S_1 \cup \{u\} \in \mathcal{F}$ . Then there exists  $u' \in S_2 \setminus S_1$  with  $(S_2 \setminus \{u'\}) \cup \{u\} \in \mathcal{B}$ .*

In order to give a better theoretical analysis of Algorithm 1 and Algorithm 2, some symbols are introduced here. For each  $j \in [B]$ ,  $s^{(j)}$  is the feasible solution of each step with  $s^{(0)} = \mathbf{0}$ . Then  $s^{(0)}, s^{(1)}, \dots, s^{(B)}$  can be obtained. The goal of the analysis for the algorithm is to analyze the relationship between the solution of algorithm  $s^{(B)}$  and an optimal solution denoted with  $\mathbf{o}$ . We define  $\mathbf{o}^{(0)}, \mathbf{o}^{(1)}, \dots, \mathbf{o}^{(B)}$  are vectors which are needed to updated in each step with  $\mathbf{o}^{(0)} = \mathbf{o}$ . According to the algorithms,  $(u^{(j)}, i^{(j)})$  is added into  $s^{(j-1)}$  to yield  $s^{(j)}$  for each  $j \in [B]$ . If  $u^{(j)} \in \text{supp}(\mathbf{o}^{(j-1)})$ , then set  $\mathbf{o}^{(j)} = u^{(j)}$ . If  $u^{(j)} \notin \text{supp}(\mathbf{o}^{(j-1)})$ , there exists  $\mathbf{o}^{(j)}$  satisfying  $(\text{supp}(\mathbf{o}^{(j-1)}) \setminus \{\mathbf{o}^{(j)}\}) \cup \{u^{(j)}\} \in \mathcal{B}$ . This can be concluded by Lemma 1. For each  $j \in [B]$ ,  $\mathbf{o}^{(j-\frac{1}{2})}$  can be constructed, which is a vector made from  $\mathbf{o}^{(j-1)}$  by

putting the  $o^{(j)}$ -th element to 0. Then  $\mathbf{o}^{(j)}$  can be made from  $\mathbf{o}^{(j-\frac{1}{2})}$  by putting  $u^{(j)}$ -th element to  $i^{(j)}$ . After the construction, we have  $\mathbf{s}^{(B)} = \mathbf{o}^{(B)}$ , which is the solution of Algorithm 1 and Algorithm 2. Moreover, we also can notice that  $\mathbf{s}^{(j-1)} \preceq \mathbf{o}^{(j-\frac{1}{2})}$  holds.

**Theorem 1.** *Assume that  $(G, \mathcal{F})$  be a matroid and  $H(x) : (k+1)^G \rightarrow \mathbb{R}^+$  is an  $\epsilon$ -approximately  $\alpha$ -weakly  $k$ -submodular set function. Algorithm 1 outputs solution  $\mathbf{s}^{(B)}$  satisfying*

$$H(\mathbf{s}^{(B)}) \geq \frac{\alpha(1-\epsilon)^2}{(1+\epsilon)[(\alpha+1)(1-\epsilon)+2B\epsilon]} H(\mathbf{o}).$$

*Proof.* In Algorithm 1,  $(u^{(j)}, i^{(j)})$  is the pair greedily selected for each  $j \in [B]$ , therefore it is the optimal pair in the step. So we have

$$\Delta_{\mathbf{o}^{(j)}, \mathbf{o}^{(j-1)}(\mathbf{o}^{(j)})} H(\mathbf{s}^{(j-1)}) \leq \Delta_{u^{(j)}, i^{(j)}} H(\mathbf{s}^{(j-1)}).$$

According to the definition of  $\epsilon$ -A- $\gamma$ -WS, we have

$$\begin{aligned} & (1-\epsilon)h(S_1^{(j-1)}, \dots, S_{\mathbf{o}^{(j-1)}(\mathbf{o}^{(j)})}^{(j-1)} \cup \{o^{(j)}\}, \dots, S_k^{(j-1)}) \\ & \leq H(S_1^{(j-1)}, \dots, S_{\mathbf{o}^{(j-1)}(\mathbf{o}^{(j)})}^{(j-1)} \cup \{o^{(j)}\}, \dots, S_k^{(j-1)}) \\ & \leq H(S_1^{(j-1)}, \dots, S_{i^{(j)}} \cup \{u^{(j)}\}, \dots, S_k^{(j-1)}) \\ & \leq (1+\epsilon)h(\mathbf{s}^{(j)}) \end{aligned}$$

Then we have

$$h(S_1^{(j-1)}, \dots, S_{\mathbf{o}^{(j-1)}(\mathbf{o}^{(j)})}^{(j-1)} \cup \{o^{(j)}\}, \dots, S_k^{(j-1)}) \leq \frac{1+\epsilon}{1-\epsilon} h(\mathbf{s}^{(j)}).$$

So the following can be obtained

$$h(S_1^{(j-1)}, \dots, S_{\mathbf{o}^{(j-1)}(\mathbf{o}^{(j)})}^{(j-1)} \cup \{o^{(j)}\}, \dots, S_k^{(j-1)}) - h(\mathbf{s}^{(j-1)}) \leq \frac{1+\epsilon}{1-\epsilon} h(\mathbf{s}^{(j)}) - h(\mathbf{s}^{(j-1)}).$$

By the definition of  $\alpha$ -weakly  $k$ -submodular set function for  $h$  and  $\mathbf{s}^{(j-1)} \preceq \mathbf{o}^{(j-\frac{1}{2})}$ , we have

$$\begin{aligned} h(\mathbf{o}^{(j-1)}) - h(\mathbf{o}^{(j)}) & \leq h(\mathbf{o}^{(j-1)}) - h(\mathbf{o}^{(j-\frac{1}{2})}) \\ & = \Delta_{\mathbf{o}^{(j)}, \mathbf{o}^{(j-1)}(\mathbf{o}^{(j)})} h(\mathbf{o}^{(j-\frac{1}{2})}) \\ & \leq \frac{1}{\alpha} \Delta_{\mathbf{o}^{(j)}, \mathbf{o}^{(j-1)}(\mathbf{o}^{(j)})} h(\mathbf{s}^{(j-1)}). \end{aligned}$$

Then we have

$$\alpha [h(\mathbf{o}^{(j-1)}) - h(\mathbf{o}^{(j)})] \leq \frac{1+\epsilon}{1-\epsilon} h(\mathbf{s}^{(j)}) - h(\mathbf{s}^{(j-1)}).$$

We can get the relationship between solutions of Algorithm 1 and the optimal solution

$$\begin{aligned}
h(\mathbf{o}) - h(\mathbf{s}^{(B)}) &= \sum_{j \in [B]} [h(\mathbf{o}^{(j-1)}) - h(\mathbf{o}^{(j)})] \\
&\leq \frac{1}{\alpha} \sum_{j \in [B]} \left[ \frac{1+\epsilon}{1-\epsilon} h(\mathbf{s}^{(j)}) - h(\mathbf{s}^{(j-1)}) \right] \\
&= \frac{1}{\alpha} \sum_{j \in [B]} \left[ \frac{1+\epsilon}{1-\epsilon} h(\mathbf{s}^{(j)}) - \frac{1+\epsilon}{1-\epsilon} h(\mathbf{s}^{(j-1)}) + \frac{1+\epsilon}{1-\epsilon} h(\mathbf{s}^{(j-1)}) - h(\mathbf{s}^{(j-1)}) \right] \\
&= \frac{1}{\alpha} \left[ \sum_{j \in [B]} \frac{1+\epsilon}{1-\epsilon} (h(\mathbf{s}^{(j)}) - h(\mathbf{s}^{(j-1)})) + \sum_{j \in [B]} \frac{2\epsilon}{1-\epsilon} h(\mathbf{s}^{(j-1)}) \right] \\
&= \frac{1}{\alpha} \left[ \frac{1+\epsilon}{1-\epsilon} (h(\mathbf{s}^{(B)}) - h(\mathbf{s}^{(0)})) + \sum_{j \in [B]} \frac{2\epsilon}{1-\epsilon} h(\mathbf{s}^{(j-1)}) \right] \\
&\leq \frac{1+\epsilon}{\alpha(1-\epsilon)} h(\mathbf{s}^{(B)}) + \frac{2B\epsilon}{\alpha(1-\epsilon)} h(\mathbf{s}^{(B)}).
\end{aligned}$$

Therefore we have

$$\begin{aligned}
h(\mathbf{o}) &\leq \left[ 1 + \frac{1-\epsilon+2B\epsilon}{\alpha(1-\epsilon)} \right] h(\mathbf{s}^{(B)}) \\
&= \frac{(\alpha+1)(1-\epsilon)+2B\epsilon}{\alpha(1-\epsilon)} h(\mathbf{s}^{(B)}).
\end{aligned}$$

We obtain

$$h(\mathbf{s}^{(B)}) \geq \frac{\alpha(1-\epsilon)}{(\alpha+1)(1-\epsilon)+2B\epsilon} h(\mathbf{o}).$$

By the definition of  $\epsilon$ -A- $\gamma$ -WS, we have

$$(1+\epsilon)H(\mathbf{s}^{(B)}) \geq \frac{\alpha(1-\epsilon)^2}{(\alpha+1)(1-\epsilon)+2B\epsilon} H(\mathbf{o}).$$

Therefore we have

$$H(\mathbf{s}^{(B)}) \geq \frac{\alpha(1-\epsilon)^2}{(1+\epsilon)[(\alpha+1)(1-\epsilon)+2B\epsilon]} H(\mathbf{o}).$$

**Theorem 2.** Assume that  $(G, \mathcal{F})$  be a matroid and  $H(x) : (k+1)^V \rightarrow \mathbb{R}^+$  is an  $\epsilon$ -approximately  $\alpha$ -weakly diminishing return. Algorithm 2 outputs solution  $\mathbf{s}^{(B)}$  satisfying

$$H(\mathbf{s}^{(B)}) \geq \frac{(1-\epsilon)\alpha}{1+\epsilon+\alpha-\epsilon\alpha} H(\mathbf{o}).$$

*Proof.* In the algorithm, for each  $j \in [B]$ ,  $(u^{(j)}, i^{(j)})$  is optimal pair selected. Then the following can be yielded

$$\Delta_{o^{(j)}, o^{(j-1)}(o^{(j)})} H(\mathbf{s}^{(j-1)}) \leq \Delta_{u^{(j)}, i^{(j)}} H(\mathbf{s}^{(j-1)}).$$

By the definition of  $\alpha$ -weakly diminishing return for  $h$ , the definition of  $\epsilon$ -approximately  $\alpha$ -weakly diminishing return for  $H$  and  $\mathbf{s}^{(j-1)} \preceq \mathbf{o}^{(j-\frac{1}{2})}$ , we have

$$\begin{aligned} \Delta_{o^{(j)}, o^{(j-1)}(o^{(j)})} H(\mathbf{o}^{(j-\frac{1}{2})}) &\leq (1 + \epsilon) \Delta_{o^{(j)}, o^{(j-1)}(o^{(j)})} h(\mathbf{o}^{(j-\frac{1}{2})}) \\ &\leq \frac{1 + \epsilon}{\alpha} \Delta_{o^{(j)}, o^{(j-1)}(o^{(j)})} h(\mathbf{s}^{(j-1)}). \end{aligned}$$

According to the introduction of  $\mathbf{o}^{(j)}$  and  $\mathbf{o}^{(j-\frac{1}{2})}$ , the following can be concluded that

$$H(\mathbf{o}^{(j-1)}) - H(\mathbf{o}^{(j)}) = \Delta_{o^{(j)}, o^{(j-1)}(o^{(j)})} H(\mathbf{o}^{(j-\frac{1}{2})}) - \Delta_{u^{(j)}, i^{(j)}} H(\mathbf{o}^{(j-\frac{1}{2})}).$$

We can get the followings

$$\begin{aligned} H(\mathbf{o}^{(j-1)}) - H(\mathbf{o}^{(j)}) &= \Delta_{o^{(j)}, o^{(j-1)}(o^{(j)})} H(\mathbf{o}^{(j-\frac{1}{2})}) - \Delta_{u^{(j)}, i^{(j)}} H(\mathbf{o}^{(j-\frac{1}{2})}) \\ &\leq \Delta_{o^{(j)}, o^{(j-1)}(o^{(j)})} H(\mathbf{o}^{(j-\frac{1}{2})}) \\ &\leq \frac{1 + \epsilon}{\alpha} \Delta_{o^{(j)}, o^{(j-1)}(o^{(j)})} h(\mathbf{s}^{(j-1)}) \\ &\leq \frac{1 + \epsilon}{\alpha(1 - \epsilon)} \Delta_{o^{(j)}, o^{(j-1)}(o^{(j)})} H(\mathbf{s}^{(j-1)}) \\ &\leq \frac{1 + \epsilon}{\alpha(1 - \epsilon)} \Delta_{u^{(j)}, i^{(j)}} H(\mathbf{s}^{(j-1)}) \\ &= \frac{1 + \epsilon}{\alpha(1 - \epsilon)} \left[ H(\mathbf{s}^{(j)}) - H(\mathbf{s}^{(j-1)}) \right] \end{aligned}$$

Therefore, we have

$$\begin{aligned} H(\mathbf{o}) - H(\mathbf{s}^{(B)}) &= \sum_{j \in [B]} \left[ H(\mathbf{o}^{(j-1)}) - H(\mathbf{o}^{(j)}) \right] \\ &\leq \frac{1 + \epsilon}{(1 - \epsilon)\alpha} \sum_{j \in [B]} \left[ H(\mathbf{s}^{(j)}) - H(\mathbf{s}^{(j-1)}) \right] \\ &= \frac{1 + \epsilon}{(1 - \epsilon)\alpha} H(\mathbf{s}^{(B)}). \end{aligned}$$

After the arrangement, we have

$$H(\mathbf{o}) \leq \left[ 1 + \frac{1 + \epsilon}{(1 - \epsilon)\alpha} \right] H(\mathbf{s}^{(B)}).$$

Therefore we can compare the values for feasible solution and the optimal solution concerning for the approximation algorithm

$$H(\mathbf{s}^{(B)}) \geq \frac{(1 - \epsilon)\alpha}{1 + \epsilon + \alpha - \epsilon\alpha} H(\mathbf{o}).$$

## 4 Discussion

In our work, we consider the approximation algorithm of maximization for non- $k$ -submodular monotone set function with noise. In the setting, a  $k$ -dimension set function with matroid constraint is investigated, which is approximately non- $k$ -submodular. The greedy technique is adapted to deal with the maximization of the set function with matroid constraint. After the considerable analysis, we obtain the approximation ratios of two kinds of situations for the set function we consider. Our conclusions generalize some known results correspondingly.

In the following works, we will discuss the topics on maximizing approximately non- $k$ -submodular monotone set function with knapsack constraint, which need to create new analysis techniques.

**Acknowledgements.** The second author is supported by the Guozhi Xu Postdoctoral Research Foundation. The third author is supported by Natural Science Foundation of China (No. 12101587) and China Postdoctoral Science Foundation (No. 2021M703167) and Fundamental Research Funds for the Central Universities (No. EIE40108X2) The fourth author is supported by Beijing Natural Science Foundation Project (No. Z200002) and National Natural Science Foundation of China (No. 12131003).

## References

1. Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., Glance, N.: Cost-effective outbreak detection in networks. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 420–429 (2007)
2. Das, A., Kempe, D.: Submodular meets spectral: greedy algorithms for subset selection, sparse approximation and dictionary selection. In: Proceedings of the 28th International Conference on Machine Learning, pp. 1057–1064 (2011)
3. Krause, A., McMahan, H.B., Guestrin, C., Gupta, A.: Robust submodular observation selection. *J. Mach. Learn. Res.* **9**, 2761–2801 (2008)
4. Kempe, D., Kleinberg, J., Tardos, É.: Maximizing the spread of influence through a social network. In: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 137–146 (2003)
5. Ohsaka, N., Yoshida, Y.: Monotone  $k$ -submodular function maximization with size constraints. In: Proceedings of the 28th International Conference on Neural Information Processing Systems, pp. 694–702 (2015)
6. Shinsaku, S.: On maximizing a monotone  $k$ -submodular function subject to a matroid constraint. *Discret. Optim.* **23**, 105–113 (2017)
7. Tang, Z., Wang, C., Chan, H.: On maximizing a monotone  $k$ -submodular function under a knapsack constraint. *Oper. Res. Lett.* **50**, 28–31 (2022)

8. Elenberg, E., Dimakis, A.G., Feldman, M., Karbasi, A.: Streaming weak submodularity: interpreting neural networks on the fly. In: Proceedings the 31st Conference on Neural Information Processing Systems, pp. 4044–4054 (2017)
9. Kuhnle, A., Smith, J.D., Crawford, V.G., Thai, M.T.: Fast maximization of non-submodular, monotonic functions on the integer lattice. In: Proceedings of the 35th International Conference on Machine Learning, pp. 2786–2795 (2018)
10. Zheng, L., Chan, H., Loukides, G., Li, M.: Maximizing approximately  $k$ -submodular functions. In: Proceedings of the 2021 SIAM International Conference on Data Mining, pp. 414–422 (2021)
11. Nguyen, L., Thai, M.T.: Streaming  $k$ -submodular maximization under noise subject to size constraint. In: Proceedings the 37th of International Conference on Machine Learning, pp. 7338–7347 (2020)
12. Horel, T., Yaron, S.: Maximization of approximately submodular functions. In: Proceedings of the 30th Conference on Neural Information Processing Systems, pp. 3053–3061 (2016)
13. Hassidim, A., Singer, Y.: Submodular optimization under noise. In: Proceedings of Conference on Learning Theory, pp. 1069–1122 (2017)
14. Matsuoka, T., Ohsaka, N.: Maximization of monotone  $k$ -submodular functions with bounded curvature and non- $k$ -submodular functions. In: Asian Conference on Machine Learning, pp. 1707–1722 (2021)
15. Ward, J., Živný, S.: Maximizing  $k$ -submodular functions and beyond. ACM Trans. Algorithms **12**, 1–26 (2016)





# Time-of-Use Scheduling Problem with Equal-Length Jobs

Vincent Chau, Chenchen Fu, Yan Lyu, Weiwei Wu, and Yizheng Zhang<sup>(✉)</sup>

School of Computer Science and Engineering, Southeast University, Nanjing, China  
yizhengzhang@seu.edu.cn

**Abstract.** In this paper, we study the scheduling problem recently introduced by Wan and Qi [NRL'2010]. We are given a set of jobs to be scheduled on a single machine, in which the cost of scheduling a job depends on when it is scheduled. This model is also known as Time-of-Use tariff. Each job is defined by its release time, its deadline and its processing time. The goal is to schedule the maximum number of jobs such that the total cost does not exceed a given budget. The problem is NP-hard when jobs have arbitrary processing time. However, when jobs have the same processing time, we show that the problem can be solved in polynomial time via dynamic programming techniques. In addition, we consider the case in which jobs have agreeable deadline, and we provide a faster algorithm.

**Keywords:** Time-of-Use · Scheduling · Dynamic programming · Uniform processing time

## 1 Introduction

Time-of-Use is a model that charges the users according to their current usage. Because the electricity production is limited, such a model has the incentive to charge the users more during peak times. Eventually, users may insist to use resources despite the price, but this may be due to the constraints of the tasks. Thus, the usage of the resource will depend on whether the activity is important. When a large number of users ask for a resource, it may lead to a crash of the system. It is particularly the case in the electricity distribution.

Such a pricing model has plenty of applications involving a demand-response-based model. No matter what the applications are, the aim is to avoid overloading a service. Examples include hotel booking, air tickets booking, cargo transportation, etc. Note that determining the pricing model is an individual research area [6].

Most of the studies in the literature sum up the cost of scheduling jobs with a performance metric (such as total completion time, lateness, etc.). Under this

---

This work is supported by the national key research and development program of China under grant N<sup>o</sup> 2019YFB2102200, and by the Fundamental Research Funds for the Central Universities N<sup>o</sup> 2242022R10024.

assumption, all the jobs need to be scheduled with the minimum cost. The user is willing to pay a certain cost for having a reasonable schedule. Each user can adjust their priority: if some activities are very important, the price for using the resources becomes negligible. On the contrary, if the user cannot afford to paying the scheduling cost during a specific period, then he is willing to wait before being served. Thus, a trade-off between quality of service and total cost needs to be determined. In this work, we consider a harder variant in which all the jobs cannot be scheduled. There are mainly two reasons: the user is given a budget (money) that he cannot exceed, or due to temporal constraints that do not allow all the jobs to be completed before their deadline. Our goal is to schedule as many jobs as possible while satisfying all the above constraints.

### 1.1 Related Works

*Cost-Aware Scheduling Problem.* This problem is referred to as the *Time Slot Cost* in the literature. We are given a set of intervals with the corresponding cost. The cost of scheduling a job is proportional to how long it is processed in each interval. If a job is scheduled during  $x$  time units during an interval of cost  $c$ , then its corresponding cost is  $x \cdot c$ . It was first studied by Wan and Qi [15] back in 2010. They investigated the Time-of-Use (TOU) cost with a traditional scheduling performance measure such as total completion time and maximum lateness. After showing that these problems are strongly NP-hard, they considered special cases on the structure of the TOU, allowing to get polynomial time algorithms. Subsequently, Zhong and Liu [18] investigated the TOU scheduling problem with the consideration of the makespan, while Kulkarni and Munagala [10] studied dynamic variants of such problems in which jobs are released over time and scheduling decisions have to be made online.

Concerning the total completion time with the ToU cost, Chen et al. [8] presented an optimal algorithm to determine the time slots to be used when the job order is predetermined for the unweighted version. They additionally designed a  $(4 + \varepsilon)$ -approximation algorithm for the weighted version. Later, Zhao et al. [17] pursued the work the total weighted completion time. They showed that when the time slot cost decreases with certain patterns, the problem can be solved in polynomial time. However, the problem becomes strongly NP-hard when the time slot cost decreases arbitrarily. Wang [16] showed that when preemption of jobs is allowed, the problem of minimizing the total TOU cost can be solved via a greedy algorithm based on matroid theory. Chen and Zhang [7] established the computational tractability of different variants of the problem. The authors provided different polynomial-time and pseudo-polynomial-time algorithms based on several assumptions about the structure of the time slot costs. The most related to our work is by Penn and Raviv [14]: they studied the problem of maximizing the (weighted) number of jobs to be scheduled minus the cost of scheduling these jobs under the TOU cost model. They proposed several polynomial-time algorithms for different cases. However, their work focuses on the case that all the jobs are released at the beginning. We extend their work by introducing release time to the jobs.

*Maximizing the Number of Jobs.* Such an objective function has been extensively studied in the literature [5]. Moore [13] gave an optimal algorithm for solving jobs with common release times. It was then extended by Lawler [12] to the case of jobs with agreeable deadlines, i.e., jobs having a later release time implies having a later deadline. When preemption of jobs is allowed (jobs can be interrupted and resumed later), Lawler [11] was the first to give a polynomial-time algorithm, and it has been further improved by Baptiste [1]. When jobs have the same processing time, Baptiste [2] was the first to show that it can be solved in polynomial time. Due to the high running time, it has been improved later on by Baptiste et al. [4]. Other results on equal lengths of jobs can be found in [3].

## 1.2 Problem Definition

We are given a set of  $n$  jobs, where each job  $j$  is characterized by its release time  $r_j$ , its deadline  $d_j$ , and its processing time  $p_j$ . Jobs must be scheduled non-preemptively on a single machine, i.e., once a job starts to be processed, it must be completed before starting another one. In this paper, we focus on the case where all jobs have the same processing time, i.e.,  $p_j = p \forall j$ .

The time horizon is divided into  $m \geq 2$  periods  $\mathcal{P} = \{P_1, \dots, P_m\}$ . Each period  $P_i$  is defined by its interval  $[a_{i-1}, a_i)$  where  $a_{i-1} < a_i$ . Moreover, each period  $P_i$  is associated with a cost  $c_i$ . A unit cost of  $c_i$  is incurred if a period  $P_i$  is used for processing a job by the machine.

The goal is to schedule as many jobs as possible while the total cost does not exceed a given budget  $B$ . We refer this problem as *throughput maximization*. Following the 3-field notation in [9], our problem can be denoted as  $1|r_j, d_j, p_j = p, c(TOU) \leq B | \sum_j U_j$ .

## 1.3 Our Contribution

Our main contribution is to show that when jobs have the same processing time, the problem of maximizing the number of jobs under a budget constraint can be solved in polynomial time. Specifically, we give the following results:

- when there is no restriction on the release times and the deadline of the jobs, we give a dynamic programming algorithm whose running time is  $O(n^6(n + m)^3)$ .
- when jobs have agreeable deadlines, meaning that a later release time implies a later deadline, we give a faster dynamic programming algorithm whose running time is  $O(n^3(n + m)^2)$ .

In Sect. 2, we give some definitions, as well as some properties that will be used throughout the paper. Then in Sect. 3, we investigate the general case while Sect. 4 focuses on the case that jobs have agreeable deadlines. Finally, we conclude in Sect. 5.

## 2 Preliminaries

On one hand, we observe that the target problem is as least as hard as the cost minimization problem. Indeed, if we can solve our problem in polynomial time, it implies that the cost minimization can be solved in polynomial time as well. This is because we can perform a binary search on the budget  $B$ , and we aim to find the smallest budget such that all the jobs are scheduled. On the other hand, when jobs have arbitrary processing time, it has been shown that the cost minimization problem is strongly NP-hard. This implies that it is also NP-hard for the throughput maximization.

Before going through structural properties, we assume without loss of generality that there is a period from 0 to  $p$  whose cost is 0. We can modify the instance by adding all the inputs by  $p$ , and it does not change the feasibility, nor the optimality of the input, since no job can be scheduled during this interval. This interval is referred to as *dummy interval*. Such a modification will be used for the design of the proposed algorithm.

Next, we focus on the structural properties of the optimal solutions. Once we get such properties, we can restrict our attention to such schedules, so that the algorithms can compute them. In the following, we delimit the schedule into small intervals. The definitions below give a first partition of the time horizon.

**Definition 1.** Let  $\Omega = \{r_j \mid j = 1, \dots, n\} \cup \{d_j \mid j = 1, \dots, n\} \cup \{a_i \mid i = 1, \dots, m\}$  be the set of time that is either a release time of a job, a deadline of a job, or a time where the price changes.

**Definition 2.** Let  $\Theta := \{t + ip \mid i = -n, \dots, n \text{ and } t \in \Omega\}$  be the set of time that a job can start or complete.

**Definition 3.** A block of jobs in a schedule is a set of maximal consecutive jobs without idle time. The length of such a block is a multiple of  $p$ .

The following proposition gives an observation on the structure of an optimal schedule.

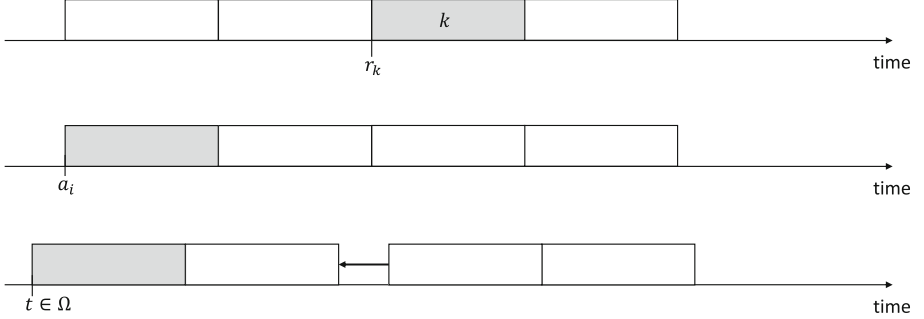
**Proposition 1.** *There exists an optimal solution in which there is at least one job starting or ending at a time in  $\Omega$  in each block of jobs.*

*Proof.* Let  $S$  be any of the optimal solutions. If  $S$  does not verify the solution, we show how to transform the solution into another one that verifies the proposition without increasing its cost. The main idea is to find a block of jobs that does not verify the proposition and to change its execution without increasing its cost.

Let  $b$  be the first block of jobs in  $S$  that does not verify the proposition. Let  $[b_s, b_e)$  denote the interval of such a block of jobs. By definition, no job in this block starts or ends in a time in  $\Omega$ , and in particular, the first job does not start at a time in  $\Omega$ , and the last job does not end at a time in  $\Omega$ , i.e.,  $b_s, b_e \notin \Omega$ .

We claim that the first time slot and the last time slot of the block of jobs have the same cost. If it is not the case, suppose that the cost of the first time slot

is more than the cost of the last time slot. Then, we can postpone the starting time of the block of jobs by one time unit. This can be done because no job in this block ends at a time in  $\Omega$ . Thus the obtained solution has a lower cost which is a contradiction with the fact that  $S$  is an optimal solution.



**Fig. 1.** Illustration of the 3 different cases of the modification. The gray job is the job that starts or ends at a time in  $\Omega$ .

The transformation is as follows. We aim to find the earliest starting time for this block of jobs as soon as possible: we push the schedule to the left-hand side until one of the following cases occurs (See Fig. 1 for an illustration):

- one of the jobs of the block starts at its release time, and it is not possible to start earlier.
- the first job (resp. last job) of the block starts (resp. ends) at a time  $a_i$ ,  $i \in \{1, \dots, m\}$ .
- the first job of the block meets the last job of the previous block of jobs, then these two blocks of jobs merge and are considered as one block of jobs. Since the previous block of jobs contains a job starting at a time in  $\Omega$ , it means that the new block does as well.

In all the above cases, the considered block of jobs verifies the claim of the Proposition. We repeat such a transformation until all the blocks of jobs on the schedule verify the Proposition.  $\square$

**Proposition 2.** *There exists an optimal solution in which jobs start or end at a time in  $\Theta$ .*

*Proof.* By combining Proposition 1 and Definition 2, the Proposition follows.  $\square$

Since preemption of jobs is not allowed, once we fix the starting time of a job, we know exactly its cost. So, we define the cost of scheduling a job from  $t$  to  $t + p$  as follows.

$$\text{cost}(t) = \sum_{z=1}^m c_z |[a_{z-1}, a_z) \cap [t, t + p)|$$

In the sequel, we only consider schedules verifying Proposition 2.

**Definition 4.** A job instance is said agreeable if jobs have agreeable deadlines, i.e., one has  $r_i \leq r_j \Leftrightarrow d_i \leq d_j$  for any pair of jobs  $i, j$ .

In the remaining of the paper, we consider that the jobs are labeled in the non-decreasing order of their deadlines, i.e.,  $d_1 \leq d_2 \leq \dots \leq d_n$ .

### 3 General Instance

In order to design the dynamic programming, we need to define the partial schedule that will be used to compose a larger schedule.

**Definition 5.** Let  $C(k, s, t, u)$  be the minimum cost of scheduling  $u$  jobs among the  $k$  first jobs released within the interval  $[s, t)$ , such that jobs are scheduled in the interval  $[s + p, t)$ .

Note that in the above definition, the interval  $[s, s + p)$  is reserved for a job. No job is currently scheduled during this interval. However, jobs released during this interval need to be considered in the schedule. The idea of the dynamic programming algorithm is to determine when the job  $k$  is scheduled. By Proposition 2, we know that there is a bounded number of starting times for jobs. So we try all the possibilities of a job's starting time. Once this is set, we can divide the schedule into two sub-schedules and finally get a recursion. Finally, we get the recursion with the following cases:

- (a) job  $k$  is not scheduled;
- (b) otherwise.

*Dynamic Programming DP1. (See Fig. 2)*

$$C(k, s, t, u) = \min \left\{ \begin{array}{l} C(k-1, s, t, u) \quad \text{(a)} \\ \min_{\substack{0 \leq u' \leq u-1 \\ t' \in \Theta \\ s \leq r_k \leq t' \leq t}} \left\{ \begin{array}{l} C(k-1, s, t', u') \\ + C(k-1, t', t, u-u'-1) \end{array} \right\} \quad \text{(b)} \end{array} \right.$$

The dynamic programming is initialized as follows.

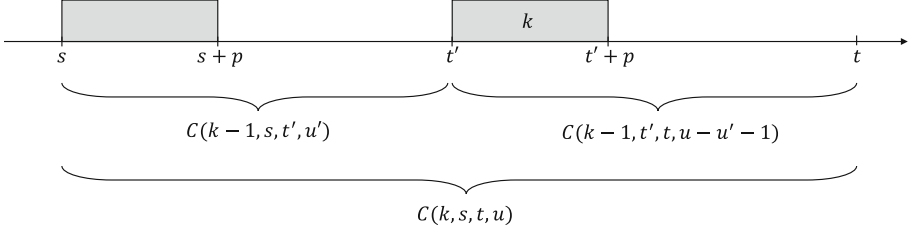
$$C(0, s, t, 0) = \begin{cases} \text{cost}(s) & \text{if } s + p \leq t, s, t \in \Theta \\ +\infty & \text{otherwise.} \end{cases}$$

**Theorem 1.** The dynamic programming DP1 computes the optimal solution.

*Proof.* The objective function is to find the maximum value  $u$  such that the total cost does not exceed the budget  $B$ . The objective function can be given as follows:

$$\arg \max_u \{C(n, 0, t, u) \leq B \mid t \in \Theta, 1 \leq u \leq n\}.$$

Recall that the interval  $[0, p)$  is a dummy interval whose cost is 0, and no job can be scheduled inside such an interval.



**Fig. 2.** Illustration of the decomposition of Dynamic Programming DP1.

The first case corresponds to the case when the job  $k$  is not scheduled, so the cost of such a schedule exactly equals  $C(k-1, s, t, u)$ . We now focus on the case that the job  $k$  is scheduled, and  $s \leq r_k \leq t$  must hold.

*Feasibility.* Fix some arbitrary time  $s \leq r_k \leq t'$  and a number of jobs  $0 < u' < u-1$ . Consider a schedule  $\mathcal{S}_1$  that realizes  $C(k-1, s, t', u')$  and another schedule  $\mathcal{S}_2$  that realizes  $C(k-1, t', t, u-u'-1)$ .

We build a schedule with  $\mathcal{S}_1$  from  $s$  to  $t'$ , with  $\mathcal{S}_2$  from  $t'$  to  $t$ , and job  $k$  scheduled within  $\mathcal{S}_2$  during the interval  $[t', t'+p]$ . Recall that by definition of  $C(k-1, t', t, u-u'-1)$ , the machine has reserved the interval  $[t', t'+p]$  and therefore does not execute any jobs. First,  $\mathcal{S}_1$  and  $\mathcal{S}_2$  select jobs from the subsets  $\{j \mid r_j \in [s, t'], j \leq k-1\}$  and  $\{j \mid r_j \in [t', t], j \leq k-1\}$  respectively. These two sets are disjoint, and their union forms the set of jobs  $\{j \mid r_j \in [s, t], j \leq k-1\}$ .

If  $r_k \in [s, t')$ , we obtain the set  $\{j \mid r_j \in [s, t], j \leq k\}$ . The above construction thus is a feasible schedule that verifies the definition of  $C(k, s, t, u)$  whose cost is at most  $C(k-1, s, t', u') + C(k-1, t', t, u-u'-1)$ .

*Optimality.* Let  $\mathcal{S}$  be the schedule that realizes  $C(k, s, t, u)$  in which the starting time of job  $k$  is maximal, i.e.,  $t'$  is maximal.

We split  $\mathcal{S}$  into two sub-schedules  $\mathcal{S}_1 \subseteq \mathcal{S}$  and  $\mathcal{S}_2 = \mathcal{S} \setminus (\mathcal{S}_1 \cup \{k\})$ . We claim that each job  $j \in \mathcal{S}_1$  starts and completes in  $[s, t')$ . Note that each job  $j$  must have a release time within this interval, i.e.,  $r_j \in [s, t')$ .

We claim that the jobs in  $\mathcal{S}_2$  are not available when job  $k$  starts (their release time is strictly greater than the starting time of job  $k$ ). This can be shown by contradiction. Suppose that there is a job  $j \in \mathcal{S}_2$  such that  $r_j \leq t'$ . It means that such a job  $j$  was available when job  $k$  starts to be executed. However, by definition of  $\mathcal{S}_1$  and  $\mathcal{S}_2$ , the job  $j$  starts after the starting time of the job  $k$ . Moreover, we know that  $j < k$  meaning that  $d_j \leq d_k$ . Therefore, the execution of jobs  $j$  and  $k$  can be swapped without any consequence on the cost and on the feasibility. Thus, we get a contradiction with the fact that the starting time of the job  $k$  is maximal.

So the restriction  $\mathcal{S}_1$  of  $\mathcal{S}$  to  $[s, t')$  is a schedule that meets all constraints related to  $C(k-1, s, t', u')$ . Hence its cost is greater than  $C(k-1, s, t', u')$ . Similarly, the restriction  $\mathcal{S}_2$  of  $\mathcal{S}$  to  $[t', t)$  is a schedule that meets all constraints related to  $C(k-1, t', t, u-u'-1)$ . To conclude the proof, note that the cost of  $\mathcal{S}$  is the sum of the costs of  $\mathcal{S}_1$  and  $\mathcal{S}_2$ .  $\square$

**Theorem 2.** *The dynamic programming DP1 has a complexity time of  $O(n^3|\Theta|^3) = O(n^6(n+m)^3)$ .*

*Proof.* The table of the dynamic programming has size of  $O(n^2|\Theta|^2)$ . Each entry of the table can be computed in time  $O(n|\Theta|)$ . Thus, the dynamic programming runs in time  $O(n^3|\Theta|^3) = O(n^6(n+m)^3)$ .  $\square$

**Corollary 1.** *When each job  $j$  is additionally associated with a weight  $w_j$ , the problem of minimizing the weighted number of late jobs can be solved in time  $O(n^4m^2(n+m)^3W^2)$  where  $W = \sum_j w_j$  is the total weight of the jobs.*

## 4 Agreeable Deadlines Jobs

This section is devoted to the case where jobs have agreeable deadline. Two jobs  $i$  and  $j$  have agreeable deadlines meaning that if  $r_i \leq r_j$ , then  $d_i \leq d_j$ , and vice versa.

**Proposition 3.** *There exists an optimal solution in which jobs are scheduled in the Earliest Deadline First (EDF) order.*

*Sketch of the Proof.* This can be proved with an exchange argument. By considering two consecutive jobs in a solution, if they are not scheduled in the EDF order, then we can swap their execution. Let the indices  $i$  and  $j$  denote these two jobs with  $i < j$ . If the job  $j$  is scheduled before job  $i$ , the exchange can be done since we have  $d_i \leq d_j$  and  $r_i \leq r_j$ , meaning that the job  $i$  can be scheduled before job  $j$  without violating their release time or deadline.

Note that the exchange argument in Proposition 3 is only valid when jobs have the same processing time. When jobs do not have the same processing time, swapping the execution of two jobs may lead to a different cost. Chen and Zhang [7] showed that for arbitrary processing time jobs, even when jobs have common release times, and common deadlines, the problem is NP-hard via a reduction from the Partition problem.

**Definition 6.** *Let  $F(k, t, u)$  be the minimum cost of scheduling  $u$  jobs among the  $k$  first jobs released before  $t$ , such that the last job is scheduled at time  $t - p$ .*

As for the general case, we distinguish two cases:

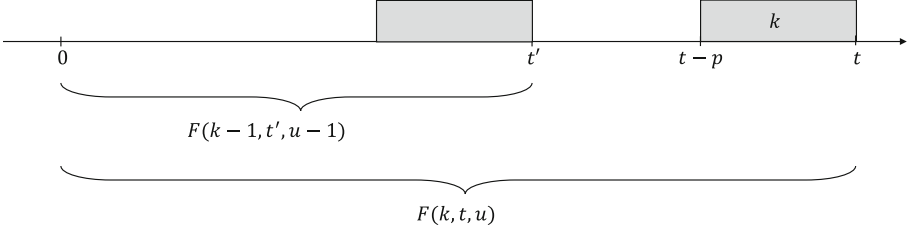
- (a) job  $k$  is not scheduled;
- (b) job  $k$  is scheduled in the interval  $[t - p, t)$ .

*Dynamic Programming DP2. (See Fig. 3)*

$$F(k, t, u) = \min \begin{cases} F(k-1, t, u), & \text{(a)} \\ \min_{\substack{t' \in \Theta \\ t' \leq t-p}} \{F(k-1, t', u-1) + \text{cost}(t-p)\} & \text{(b)} \end{cases}$$

The initialization of the dynamic programming is presented as follows.





**Fig. 3.** Illustration of the decomposition of Dynamic Programming DP2 for jobs with agreeable deadlines

$$F(0, t, u) = \min \begin{cases} 0 & \text{if } (t = 0 \text{ or } t = p) \text{ and } u = 0, \\ +\infty & \text{otherwise.} \end{cases}$$

Recall that there is a dummy period in  $[0, p)$  whose cost is 0, so we can suppose that this interval is used during the initialization.

**Theorem 3.** *The dynamic programming DP2 computes an optimal solution.*

*Proof.* The objective function is

$$\arg \max_u \{F(n, t, u) \leq B \mid t \in \Theta, 1 \leq u \leq n\}.$$

Let  $\mathcal{S}$  be the schedule that realizes  $F(k, t, u)$ . If job  $k$  is not scheduled in  $\mathcal{S}$ , then its cost is exactly  $F(k-1, t, u)$ . In the remaining of the proof, we focus on the case that job  $k$  is scheduled in  $\mathcal{S}$ , and it is scheduled in  $[t-p, t)$ .

*Feasibility.* We first discuss that the constructed solution is feasible. Fix a value  $t' \leq t-p$ , and  $t' \in \Theta$ , and let  $\mathcal{S}'$  be a schedule that realizes  $F(k-1, t', u-1)$ . We construct a schedule from 0 to  $t-p$  with  $\mathcal{S}'$ , and with job  $k$  from  $t-p$  to  $p$ . Jobs in  $\mathcal{S}'$  are chosen from the  $k-1$  first jobs. By adding job  $k$ , the schedule  $\mathcal{S}$  chooses from the  $k$  first jobs, which corresponds to the definition of  $F(k, t, u)$ . Finally, we test all the possible completion times of the sub-schedules less than  $t-p$ , so the constructed solution, whose cost is no more than  $F(k-1, t', u-1) + \text{cost}(t-p)$ , is feasible.

*Optimality.* Let  $\mathcal{S}'$  be the sub-schedule of  $\mathcal{S}$  in the interval  $[0, t')$  such that the machine is idle in  $[t', t-p)$ . By Proposition 3, the jobs in  $\mathcal{S}'$  should be completed before job  $k$  starts, i.e., before  $t-p$ . Since the machine is idle in  $[t', t-p)$ , then we know that there is a job scheduled in  $[t'-p, t')$ . So the restriction  $\mathcal{S}'$  to  $\mathcal{S}$  is a schedule that meets all constraints related to  $F(k-1, t', u-1)$ . Thus, the cost of such a schedule is the sum of the cost of  $\mathcal{S}'$  and the additional cost  $\text{cost}(t-p)$  for scheduling job  $k$ .  $\square$

**Theorem 4.** *The dynamic programming DP2 has a complexity time of  $O(n^3(n+m)^2)$ .*

*Proof.* The table of the dynamic programming has size of  $O(n^2|\Theta|)$ . Each entry of the table can be computed in time  $O(|\Theta|)$ . Thus, the dynamic programming runs in time  $O(n^2|\Theta|^2) = O(n^3(n+m)^2)$ .  $\square$

**Corollary 2.** *When each job  $j$  is additionally associated with a weight  $w_j$ , the problem of minimizing the weighted number of late jobs can be solved in time  $O(n^2(n+m)^2W)$  where  $W = \sum_j w_j$  is the total weight of the jobs.*

## 5 Conclusion

In this work, we showed that when jobs have the same processing time, maximizing the number of jobs that can be scheduled under a budget constraint can be solved in polynomial time. The proposed algorithms can be extended to the weighted variant and get pseudo-polynomial-time algorithms, from which FPTASs can be derived. Whether the weighted version can be solved in polynomial time is still unknown and would be of great interest for future work.

## References

1. Baptiste, P.: An  $o(n^4)$  algorithm for preemptive scheduling of a single machine to minimize the number of late jobs. *Oper. Res. Lett.* **24**(4), 175–180 (1999)
2. Baptiste, P.: Polynomial time algorithms for minimizing the weighted number of late jobs on a single machine with equal processing times. *J. Scheduling* **2**(6), 245–252 (1999)
3. Baptiste, P., Brucker, P., Knust, S., Timkovsky, V.G.: Ten notes on equal-processing-time scheduling. *4OR* **2**(2), 111–127 (2004)
4. Baptiste, P., Chrobak, M., Dürr, C., Jawor, W., Vakhania, N.: Preemptive scheduling of equal-length jobs to maximize weighted throughput. *Oper. Res. Lett.* **32**(3), 258–264 (2004)
5. Brucker, P.: *Scheduling Algorithms*, 5th edn. Springer Publishing Company, Incorporated (2010)
6. Chawla, S., Devanur, N.R., Holroyd, A.E., Karlin, A.R., Martin, J.B., Sivan, B.: Stability of service under time-of-use pricing. In: Hatami, H., McKenzie, P., King, V. (eds.) *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19–23, 2017*, pp. 184–197. ACM (2017)
7. Chen, B., Zhang, X.: Scheduling with time-of-use costs. *Eur. J. Oper. Res.* **274**(3), 900–908 (2019)
8. Chen, L., Megow, N., Rischke, R., Stougie, L., Verschae, J.: Optimal algorithms for scheduling under time-of-use tariffs. *Ann. Oper. Res.* **304**(1), 85–107 (2021)
9. Graham, R.L., Lawler, E.L., Lenstra, J.K., Kan, A.: Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annal. Discrete Math.* **5**, 287–326 (1979)
10. Kulkarni, J., Munagala, K.: Algorithms for cost-aware scheduling. In: Erlebach, T., Persiano, G. (eds.) *WAOA 2012. LNCS*, vol. 7846, pp. 201–214. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-38016-7\\_17](https://doi.org/10.1007/978-3-642-38016-7_17)

11. Lawler, E.: A dynamic programming algorithm for preemptive scheduling of a single machine to minimize the number of late jobs, vol. 26, pp. 125–133. Baltzer Science Publishers, Baarn/Kluwer Academic Publishers (1990)
12. Lawler, E.: Knapsack-like scheduling problems, the moore-hodgson algorithm and the ‘tower of sets’ property, vol. 20, pp. 91–106 (1994)
13. Moore, J.M.: An  $n$  job, one machine sequencing algorithm for minimizing the number of late jobs, vol. 15, pp. 102–109, September 1968
14. Penn, M., Raviv, T.: Complexity and algorithms for min cost and max profit scheduling under time-of-use electricity tariffs. *J. Sched.* **24**(1), 83–102 (2021)
15. Wan, G., Qi, X.: Scheduling with variable time slot costs. *Naval Res. Logist. (NRL)* **57**(2), 159–171 (2010)
16. Wang, K.: Calibration scheduling with time slot cost. *Theor. Comput. Sci.* **821**, 1–14 (2020)
17. Zhao, Y., Qi, X., Li, M.: On scheduling with non-increasing time slot cost to minimize total weighted completion time. *J. Sched.* **19**(6), 759–767 (2016)
18. Zhong, W., Liu, X.: A single machine scheduling problem with time slot costs. In: Qian, Z., Cao, L., Su, W., Wang, T., Yang, H. (eds.) *Recent Advances in Computer Science and Information Engineering. Lecture Notes in Electrical Engineering*, vol. 126, pp. 677–681. Springer, Heidelberg (2012)



# Online Weakly DR-Submodular Optimization with Stochastic Long-Term Constraints

Junkai Feng<sup>1</sup>, Ruiqi Yang<sup>1,2(✉)</sup>, Yapu Zhang<sup>1</sup>, and Zhenning Zhang<sup>1</sup>

<sup>1</sup> Beijing Institute for Scientific and Engineering Computing, Beijing University of Technology, Beijing 100124, People's Republic of China

<sup>2</sup> School of Mathematical Sciences, University of Chinese Academy Sciences, Beijing 100049, People's Republic of China

yangruiqi@ucas.ac.cn

**Abstract.** The online optimization has been extensively studied under a variety of different settings. In this paper, we consider the online maximization problems with stochastic linear cumulative constraints, where the objective functions are the sum of  $\rho$ -weakly DR-submodular functions and concave functions. Inspired by the penalty function strategy, we propose an algorithm of primal-dual type to solve this class of problems. Under mild conditions, we show that the algorithm achieves sub-linear regret bounds and cumulative budget violation bounds with high probability.

**Keywords:** Online maximization · Weakly DR-submodular · Concave · Sublinear

## 1 Introduction

In the era of big data, many pieces of information arrives at any time. In many fields, such as artificial intelligence, machine learning, etc., firstly the learner need to specify action with uncertain future information, and then get feedback related to this action in hindsight. Based on the adding new experience, the learner make next decision again. Repeatedly do it many rounds, this process is just online optimization. Specifically, at each round  $t \in [T]$ , after deciding the action  $x_t$  from the known domain set  $\mathcal{X}$ , the corresponding objective utility function  $f_t$  (and constraint function  $g_t$ ) can be revealed. The goal is to optimize the overall utility, that is minimizing the  $\rho$ -regret:

$$\rho \max_{x \in \mathcal{X} \cap \mathcal{Q}} \sum_{t=1}^T f_t(x) - \sum_{t=1}^T f_t(x_t),$$

where  $\mathcal{Q}$  represents certain accumulative limitation on the decision sequences  $\{x_t\}_{t \in [T]}$ , i.e.,  $\{x : \sum_{t=1}^T g_t(x) \leq 0\}$ . See [1, 2] for online optimization with constraints over the accumulated decisions. Generally speaking, the online optimization is being studied from two aspects: convex and nonconvex.

For the online convex optimization (OCO), 1-regret is considered. Without long term constraints, OCO has been studied exhaustively (see references [3–5]). If the objective functions and constraint functions are chosen adversely, Mannor, Tsitsiklis, and Yu [6] show that: no matter which actions are chosen, the regret can not be guaranteed sublinear in  $T$  while the overall budget violation is sublinear too. Thus, in order to analyze useful results, some researchers propose using a new weaker comparator in the regret, that is,

$$\mathcal{Q}' = \{x : \sum_{t=i}^{i+W-1} g_t(x) \leq 0, \forall 1 \leq i \leq T - W + 1\},$$

see [7–9] for detailed performance results.

Consider the online nonconvex optimization in which the utility functions are DR-submodular. Based on the Frank-Wolfe variant proposed in [10], Chen [11] proposed the online Meta-Frank-Wolfe for maximizing unconstrained monotone DR-submodular functions, more related results see [12–16]. For different settings, deterministically or stochastically, fixedly or adversely, corresponding results have been shown under DR-submodular framework, see [10, 17–19].

Inspired by the work in [19] and [10], we consider the online neither convex nor DR-submodular optimization problems with stochastic cumulative constraints, where the objective function is the sum of weakly DR-submodular functions and concave functions. An algorithm of primal-dual type is proposed, and the sublinear performance is analyzed in probabilistic sense.

## 2 Preliminaries

**Notations.** Given  $m \in \mathbb{N}$ ,  $[m]$  denotes the index set  $\{1, 2, \dots, m\}$ . For any  $x, y \in \mathcal{R}^n$ ,  $x \preceq y$  means that  $x_i \leq y_i, \forall i \in [n]$ . The join(meet) operation  $\vee(\wedge)$  operation between two vectors  $x, y \in \mathcal{R}^n$  means that

$$(x \vee y)_i = \max\{x_i, y_i\} ((x \wedge y)_i = \min\{x_i, y_i\}), \forall i \in [n].$$

Given  $S \subseteq \mathcal{R}^n$ , the function  $f : S \rightarrow \mathcal{R}$  is said to be monotone on  $S$ , if  $f(x) \leq f(y), \forall x, y \in S$  with  $x \preceq y$ . In addition, if  $S$  is closed and convex,  $\mathcal{P}_S$  denotes the projection operator onto  $S$ . It's well known that  $\mathcal{P}_S$  is single valued and firmly nonexpansive. For  $a \in \mathcal{R}$ , we use  $[a]_+$  to denote the projection of  $a$  onto  $\mathcal{R}_+$ , i.e.,  $[a]_+ := \mathcal{P}_{\mathcal{R}_+}(a) = \max\{a, 0\}$ .

**Weakly DR-Submodular Ratio.** Given  $S \subseteq \mathcal{R}^n$ , let  $f : S \rightarrow \mathcal{R}$  be a differentiable function on  $S$ .  $f$  is called DR-submodular on  $S$ , if  $\forall x, y \in S$  with  $x \succeq y$ , it has

$$\nabla f(x) \preceq \nabla f(y).$$

In addition, if  $f$  is monotone, set

$$\rho := \inf_{\substack{x \succeq y \\ x, y \in S}} \inf_{i \in [n]} \frac{\nabla_i f(x)}{\nabla_i f(y)}.$$

Obviously,  $0 \leq \rho \leq 1$  and  $\nabla f(x) \succeq \rho \nabla f(y)$ ,  $\forall x \preceq y$ . And  $f$  is said to be  $\rho$ -weakly DR-submodular (see [20]). For weakly DR-submodular and concave functions, we show the following relation of inequality.

**Lemma 1.** *Given  $\mathcal{X} \subseteq \mathcal{R}_+^n$ ,  $f, \theta : \mathcal{X} \rightarrow \mathcal{R}$  are two monotone and differentiable functions. Moreover, they are  $\rho$ -weakly DR-submodular and concave, respectively. Set  $F = f + \theta$ , then for any  $x, y \in \mathcal{X}$ , it holds*

$$\langle \nabla F(x), y \rangle \geq \rho(F(y) - F(x)).$$

*Proof.* According to the monotonicity of  $f$ , we obtain that

$$\begin{aligned} \rho(f(y) - f(x)) &\leq \rho(f(y \vee x) - f(x)) \\ &\leq \langle \nabla f(x), y \vee x - x \rangle \\ &= \langle \nabla f(x), y - y \wedge x \rangle \\ &\leq \langle \nabla f(x), y \rangle, \end{aligned}$$

where the second inequality comes from the following property of  $\rho$ -weakly DR-submodular function  $f$ ,

$$\rho(f(y) - f(x)) \leq \langle \nabla f(x), y - x \rangle, \quad \forall y \succeq x \text{ or } y \preceq x.$$

It follows from the monotonicity and concavity of  $\theta$  that

$$\rho(\theta(y) - \theta(x)) \leq \rho \langle \nabla \theta(x), y - x \rangle \leq \langle \nabla \theta(x), y \rangle.$$

Thus, we get the conclusion of the Lemma.

### 3 Problem Statement

**The Considered Problem.** In this article, we consider the following online optimization problem with stochastic cumulative constraint,

$$\begin{aligned} \max \quad & \sum_{t=1}^T F_t(x_t) \\ \text{s.t.} \quad & \sum_{t=1}^T \langle p, x_t \rangle \leq B, \\ & x_t \in \mathcal{X}, \quad \forall t \in [T], \end{aligned} \tag{1}$$

where  $\mathcal{X} \subseteq \mathcal{R}_+^n$  is a nonempty, compact, and convex set,  $0 \in \mathcal{X}$ ,  $F_t : \mathcal{X} \rightarrow \mathcal{R}_+$  ( $t \in [T]$ ) are normalized (i.e.,  $F_t(0) = 0$ ) utility functions,  $B > 0$  is the limited budget, and  $p \succeq 0$ . Specifically, in each round  $t \in [T]$ , based on the information of experience, the operator firstly chooses an action  $x_t \in \mathcal{X}$ , then the utility function  $F_t$  and the random constraint vector  $p_t \sim \mathcal{D}(p, \mathcal{C})$  are revealed. Here, the random vectors  $\{p_t\}_{t \in [T]}$  are independent identically distributed, and its mean  $p$  and covariance  $\mathcal{C}$  are unknown. Throughout this paper, we make the following additional assumptions on problem (1).

**Assumption 1.** (A1) For each  $t \in [T]$ ,  $F_t(\cdot) = f_t(\cdot) + \theta_t(\cdot)$ , where  $f_t, \theta_t : \mathcal{X} \rightarrow \mathcal{R}_+$  are both monotone and  $L_t$  smooth. Moreover,  $f_t, \theta_t$  are  $\rho$ -weakly DR-submodular and concave, respectively.

(A2) The random vectors  $\{p_t\}_{t \in [T]}$  are sampled over a nonnegative bounded set.

The above conditions imply the following facts.

*Remark 1.* (1) Set  $L := 2 \max_{t \in [T]} L_t < \infty$ , then  $F_t(t \in [T])$  are all  $L$  smooth.

(2) The diameter of  $\mathcal{X}$  is a finite number, that is,

$$d := \max_{x, y \in \mathcal{X}} \|x - y\| = \max_{x \in \mathcal{X}} \|x\| < \infty.$$

(3) Since  $\mathcal{X}$  is bounded,  $F_t$  is Lipschitz on  $\mathcal{X}$ . Together with Assumption 1 (A2), we can take  $0 < \delta < \infty$ , such that for any  $t \in [T]$ ,  $x, y \in \mathcal{X}$ ,

$$\Pr\{\|p_t\| \leq \delta\} = 1, \quad |F_t(x) - F_t(y)| \leq \delta \|x - y\|.$$

(4) According to Assumption 1 (A2),

$$U := \max_{q \sim \mathcal{D}(p, \mathcal{C}), x \in \mathcal{X}} \left| \langle q, x \rangle - \frac{B}{T} \right| \leq \max\{\delta d - \frac{B}{T}, \frac{B}{T}\} < \infty.$$

**Performance Metric.** After implementing an algorithm, we get a sequence of actions  $\{x_t\}_{t \in [T]}$ . In order to evaluate the quality of the algorithm, we introduce two notations:  $(1 - \frac{1}{e^\rho})$ -regret and the long term constraint violation.

**Definition 1.** Chosen a sequence of actions  $\{x_t\}_{t \in [T]}$ .

(1) The  $(1 - \frac{1}{e^\rho})$ -regret is defined as:

$$\mathcal{R}_T := \left(1 - \frac{1}{e^\rho}\right) \max_{x \in \mathcal{X}^*} \sum_{t=1}^T F_t(x) - \sum_{t=1}^T F_t(x_t),$$

where  $\mathcal{X}^* = \{x \in \mathcal{X} : \sum_{t=1}^T \langle p, x \rangle \leq B\} = \{x \in \mathcal{X} : \langle p, x \rangle \leq \frac{B}{T}\}$ .

(2) The cumulative constraint violation is defined as:

$$\mathcal{C}_T := \sum_{t=1}^T \langle p, x_t \rangle - B.$$

The regret measures the gap between the total reward of best fixed action multiplied by  $(1 - \frac{1}{e^\rho})$  in hindsight and that of the algorithm, while the violation describes how much the resource consumption incurred by the algorithm exceeded the budget.

## 4 Algorithm and Performance Analysis

In this section, we propose our online method of primal-dual type to solve problem (1) and analyze its performance.

---

**Algorithm** Online algorithm of primal-dual type.

---

**Input:** Constraint set  $\mathcal{X}$ , time horizon  $T$ , positive integer  $K$ , stepsize  $\alpha > 0$ , penalty parameter  $\beta := \delta^2\alpha$ , and nonnegative parameter sequence  $\{\eta_t\}_{t \in [T]}$ .

**Output:**  $\{x_t\}_{t \in [T]}$ .

Initialize  $F_0(x) = 0, \forall x \in \mathcal{X}, \lambda_0 = \tilde{p}_0 = 0, x_0^{(k)} = v_0^{(k)} = 0, \forall k \in [K]$ .

**for**  $t = 1$  **to**  $T$  **do**

$x_t^{(1)} = 0$ .

**for**  $k = 1$  **to**  $K$  **do**

$$v_t^{(k)} = \mathcal{P}_{\mathcal{X}}(v_{t-1}^{(k)} + \alpha(\nabla F_{t-1}(x_{t-1}^{(k)}) - \lambda_{t-1}\tilde{p}_{t-1})), \quad (2)$$

$$x_t^{(k+1)} = x_t^{(k)} + \frac{1}{K}v_t^{(k)}. \quad (3)$$

**end for**

Set  $x_t = x_t^{(K+1)}$  and act  $x_t$ .

Observe the utility function  $F_t(\cdot) = f_t(\cdot) + \theta_t(\cdot)$  and the random constraint vector sampled as  $p_t$ . Set  $\tilde{p}_t = \frac{1}{t} \sum_{i=1}^t p_i$ , and  $g_t(\cdot) = \langle \tilde{p}_t, \cdot \rangle - \frac{\beta}{T}$ ,  $h_t(\cdot) = g_t(\cdot) - \eta_t$ .

Compute  $\lambda_t = \frac{1}{\beta}[h_t(x_t)]_+$

**end for**

---

*Remark 2.* After executing the algorithm, all the utility functions  $\{F_t = f_t + \theta_t\}_{t \in [T]}$  and random constraint vectors  $\{p_t\}_{t \in [T]}$  are revealed. For convenience, define  $\mathcal{L}_t(\cdot, \cdot) : \mathcal{R}^n \times \mathcal{R} \rightarrow \mathcal{R}$  as

$$\mathcal{L}_t(x, \lambda) = f_t(x) + \theta_t(x) + \frac{\beta}{2}\lambda^2 - \lambda h_t(x).$$

Then  $\nabla_x \mathcal{L}_t(x, \lambda) = \nabla F_t(x) - \lambda \tilde{p}_t$ , and hence for fixed  $\lambda$ ,  $\mathcal{L}_t(x, \lambda)$  is  $L$  smooth with respect to  $x$  variable. Observe that the iterative formula (2) is just

$$v_t^{(k)} = \mathcal{P}_{\mathcal{X}}(v_{t-1}^{(k)} + \alpha \nabla_x \mathcal{L}_{t-1}(x_{t-1}^{(k)}, \lambda_{t-1})). \quad (4)$$

Taking  $x \in \mathcal{X}$ , since  $\mathcal{P}_{\mathcal{X}}$  is nonexpansive, we can deduce that, for each  $t \in [T]$ ,

$$\begin{aligned} & 2\alpha \langle \nabla_x \mathcal{L}_t(x_t^{(k)}, \lambda_t), x - v_t^{(k)} \rangle \\ \leq & \|v_t^{(k)} - x\|^2 - \|v_{t+1}^{(k)} - x\|^2 + \alpha^2 \|\nabla_x \mathcal{L}_t(x_t^{(k)}, \lambda_t)\|^2. \end{aligned}$$



Therefore

$$\begin{aligned}
 & \sum_{t=1}^T \langle \nabla_x \mathcal{L}_t(x_t^{(k)}, \lambda_t), x - v_t^{(k)} \rangle \\
 & \leq \frac{1}{2\alpha} \|v_1^{(k)} - x\|^2 + \frac{\alpha}{2} \sum_{t=1}^T \|\nabla_x \mathcal{L}_t(x_t^{(k)}, \lambda_t)\|^2 \\
 & \leq \frac{d^2}{2\alpha} + \frac{\alpha}{2} \sum_{t=1}^T (2\delta^2 + 2\delta^2 \lambda_t^2) \\
 & = \frac{d^2}{2\alpha} + \alpha T \delta^2 + \alpha \delta^2 \sum_{t=1}^T \lambda_t^2.
 \end{aligned} \tag{5}$$

For simplicity, denote  $g(\cdot) = \langle p, \cdot \rangle - \frac{B}{T}$  in the following.

**Lemma 2.** *Let  $\{x_t\}_{t \in [T]}$  be the sequence generated by the proposed algorithm. Then, for any  $x \in \mathcal{X}$ , we have*

$$\begin{aligned}
 & \left(1 - \frac{1}{e^\rho}\right) \sum_{t=1}^T F_t(x) - \sum_{t=1}^T F_t(x_t) \\
 & \leq \sum_{t=1}^T \lambda_t h_t(x) + \frac{d^2}{2\alpha} + \alpha \delta^2 T + \frac{L d^2}{2K} T.
 \end{aligned} \tag{6}$$

*Proof.* Since  $\mathcal{L}_t(\cdot, \lambda)$  is  $L$  smooth, it follows from the Descent Lemma that,  $\forall x \in \mathcal{X}, k \in [K]$ ,

$$\begin{aligned}
 & \mathcal{L}_t(x_t^{(k+1)}, \lambda_t) \\
 & \geq \mathcal{L}_t(x_t^{(k)}, \lambda_t) + \langle \nabla_x \mathcal{L}_t(x_t^{(k)}, \lambda_t), x_t^{(k+1)} - x_t^{(k)} \rangle - \frac{L}{2} \|x_t^{(k+1)} - x_t^{(k)}\|^2 \\
 & = \mathcal{L}_t(x_t^{(k)}, \lambda_t) + \frac{1}{K} \langle \nabla_x \mathcal{L}_t(x_t^{(k)}, \lambda_t), v_t^{(k)} \rangle - \frac{L}{2K^2} \|v_t^{(k)}\|^2 \\
 & = \mathcal{L}_t(x_t^{(k)}, \lambda_t) + \frac{1}{K} \langle \nabla F_t(x_t^{(k)}), x \rangle - \frac{1}{K} \langle \lambda_t \tilde{p}_t, x \rangle - \frac{L}{2K^2} \|v_t^{(k)}\|^2 \\
 & \quad + \frac{1}{K} \langle \nabla_x \mathcal{L}_t(x_t^{(k)}, \lambda_t), v_t^{(k)} - x \rangle \\
 & \geq \mathcal{L}_t(x_t^{(k)}, \lambda_t) + \frac{\rho}{K} (F_t(x) - F_t(x_t^{(k)})) - \frac{1}{K} \langle \lambda_t \tilde{p}_t, x \rangle - \frac{L}{2K^2} d^2 \\
 & \quad + \frac{1}{K} \langle \nabla_x \mathcal{L}_t(x_t^{(k)}, \lambda_t), v_t^{(k)} - x \rangle,
 \end{aligned}$$

where the first equality and last inequality comes from formula (3) and Lemma 1, respectively. Based on the definition of  $\mathcal{L}_t$  and iterative formula (3), the above

inequality can be reformulated as

$$F_t(x) - F_t(x_t^{(k+1)}) \leq (1 - \frac{\rho}{K})(F_t(x) - F_t(x_t^{(k)})) + \frac{1}{K}(\langle \nabla_x \mathcal{L}_t(x_t^{(k)}), \lambda_t \rangle, x - v_t^{(k)}) \\ + \langle \lambda_t \tilde{p}_t, x - v_t^{(k)} \rangle + \frac{L}{2K}d^2.$$

By recursive calculation, one can find that

$$F_t(x) - F_t(x_t^{(K+1)}) \\ \leq \frac{1}{K} \cdot \sum_{k=1}^K (1 - \frac{\rho}{K})^{K-k} (\langle \nabla_x \mathcal{L}_t(x_t^{(k)}), \lambda_t \rangle, x - v_t^{(k)}) + \langle \lambda_t \tilde{p}_t, x - v_t^{(k)} \rangle + \frac{L}{2K}d^2 \\ + (1 - \frac{\rho}{K})^K (F_t(x) - F_t(x_t^{(1)})).$$

Since  $(1 - \frac{\rho}{K})^K \leq \frac{1}{e^\rho}$ , summing the above inequality over  $t = 1, 2, \dots, T$ , we get that

$$(1 - \frac{1}{e^\rho}) \sum_{t=1}^T F_t(x) - \sum_{t=1}^T F_t(x_t) \\ \leq \frac{1}{K} \sum_{t=1}^T \sum_{k=1}^K (1 - \frac{\rho}{K})^{K-k} (\langle \nabla_x \mathcal{L}_t(x_t^{(k)}), \lambda_t \rangle, x - v_t^{(k)}) \\ + \langle \lambda_t \tilde{p}_t, x - v_t^{(k)} \rangle + \frac{L}{2K}d^2. \quad (7)$$

By the updating rule of  $x_t^{(k)}$  and  $\lambda_t$ , we deduce that

$$\frac{1}{K} \sum_{t=1}^T \sum_{k=1}^K \langle \lambda_t \tilde{p}_t, x - v_t^{(k)} \rangle = \sum_{t=1}^T \langle \lambda_t \tilde{p}_t, x - x_t \rangle \\ = \sum_{t=1}^T \lambda_t [h_t(x) - h_t(x_t)],$$

and  $\alpha\delta^2\lambda_t^2 - \lambda_t h_t(x_t) = 0$ . Thus, combining with formulas (5) and (7), we get the conclusion.

**Lemma 3.** For fixed  $\varepsilon > 0$ , take  $\eta_t = U\sqrt{\frac{2\ln\frac{2T}{\varepsilon}}{t}}, \forall t \in [T]$ . For any  $x \in \mathcal{X}$ , consider a sequence of events  $\mathcal{E}_t := \{h_t(x) \leq g(x)\}, t \in [T]$ . Then

$$(1) \Pr(\mathcal{E}_t) \geq 1 - \frac{\varepsilon}{T}, \forall t \in [T],$$

$$(2) \Pr(\bigcap_{t \in [T]} \mathcal{E}_t) \geq 1 - \varepsilon.$$

*Proof.* (1) Note that  $\mathbb{E}[\langle p_i, x \rangle - \frac{B}{T}] = g(x), \forall i \in [T]$  and

$$g_t(x) = \langle \tilde{p}_t, x \rangle - \frac{B}{T} = \frac{1}{t} \sum_{i=1}^t (\langle p_i, x \rangle - \frac{B}{T}),$$

we conclude from Remark 1 (4) and Hoeffding's inequality that

$$\Pr\{|g_t(x) - g(x)| \geq \eta_t\} \leq 2 \exp(-\frac{2t^2\eta_t^2}{4tU^2}) = \frac{\varepsilon}{T}.$$

Thus,

$$\Pr(\mathcal{E}_t) \geq \Pr\{|g_t(x) - g(x)| \leq \eta_t\} \geq 1 - \frac{\varepsilon}{T}.$$

(2) It follows from the result of (1) that

$$\begin{aligned} \Pr\left(\bigcap_{t \in [T]} \mathcal{E}_t\right) &= 1 - \Pr\left(\bigcup_{t \in [T]} \mathcal{E}_t^c\right) \\ &\geq 1 - \sum_{t \in [T]} \Pr(\mathcal{E}_t^c) \\ &\geq 1 - \varepsilon. \end{aligned}$$

**Theorem 1.** *Given  $0 < \varepsilon < 1$ , let  $\{\eta_t\}_{t \in [T]}$  be chosen as in Lemma 3. If  $\alpha = \frac{\sqrt{2d}}{2\delta\sqrt{T}}$ ,  $K = \sqrt{T}$ , then*

$$\Pr\{\mathcal{R}_T \leq (\sqrt{2}\delta + \frac{Ld}{2})d\sqrt{T}\} \geq 1 - \varepsilon.$$

*Proof.* Take  $x^* \in \mathcal{X}^*$ , then  $g(x^*) \leq 0$ . By Lemma 3 (2), we obtain that

$$\begin{aligned} \Pr\left\{\sum_{t \in [T]} \lambda_t h_t(x^*) \leq 0\right\} &\geq \Pr\left(\bigcap_{t \in [T]} \{h_t(x^*) \leq 0\}\right) \\ &\geq \Pr\left(\bigcap_{t \in [T]} \{h_t(x^*) \leq g(x^*)\}\right) \\ &\geq 1 - \varepsilon. \end{aligned}$$

Hence, setting  $x = x^*$  in formula (6), and substituting  $\alpha = \frac{\sqrt{2d}}{2\delta\sqrt{T}}$ ,  $K = \sqrt{T}$  into it, we get the conclusion.

**Lemma 4.** *There exists  $C > 0$ , such that the following holds: for any  $0 < \varepsilon < 1$ ,*

$$\Pr\left\{\sum_{t=1}^T \|p - \tilde{p}_t\| \leq 2C\delta\sqrt{(T+1)\ln\frac{2nT}{\varepsilon}}\right\} \geq 1 - \varepsilon.$$

*Proof.* Since  $\{p_i\}_{i \in [T]}$  are i.i.d. and drawn from bounded set, there exists  $C_1 > 0$ , such that  $\forall a \in \mathcal{R}, i \in [T]$ ,

$$\Pr\{\|p_i - p\| \geq a\} \leq 2 \exp\left(-\frac{a^2}{2C_1^2\delta^2}\right).$$

Thus, by Corollary 7 in [21], there exists  $C_2 > 0$ , such that  $\forall t \in [T]$ ,

$$\Pr\left\{\left\|\sum_{i \in [t]} (p_i - p)\right\| \leq C_2 \sqrt{\sum_{i \in [t]} (C_1^2\delta^2) \ln \frac{2nT}{\varepsilon}}\right\} \geq 1 - \frac{\varepsilon}{T},$$

that is

$$\Pr\left\{\|\tilde{p}_t - p\| \leq \frac{C_1 C_2 \delta}{\sqrt{t}} \sqrt{\frac{2nT}{\varepsilon}}\right\} \geq 1 - \frac{\varepsilon}{T}.$$

Set  $C := C_1 C_2$ . Note that  $\sum_{t \in [T]} \frac{1}{\sqrt{t}} \leq 2\sqrt{T+1}$ , using the same technique as in Lemma 3, we get the conclusion.

**Theorem 2.** Given  $0 < \varepsilon < 1$ , let  $\{\eta_t\}_{t \in [T]}, \alpha, K$  take the same values as in Theorem 1. Then the probability of the following event is at least  $1 - \varepsilon$ ,

$$\begin{aligned} \mathcal{C}_T &\leq 2C\delta d \sqrt{(T+1) \ln \frac{2nT}{\varepsilon}} + 2U \sqrt{2(T+1) \ln \frac{2T}{\varepsilon}} \\ &\quad + \frac{\sqrt{2}d\delta^2}{2B} T \sqrt{T} + \frac{\sqrt{2}L\delta d^3}{4B} T + \frac{\delta^2 d^2}{B} T. \end{aligned}$$

*Proof.* According to the definition of  $\mathcal{C}_T$ , we have

$$\begin{aligned} \mathcal{C}_T &= \sum_{t=1}^T \langle p - \tilde{p}_t, x_t \rangle + \sum_{t=1}^T h_t(x_t) + \sum_{t=1}^T \eta_t \\ &\leq d \sum_{t=1}^T \|p - \tilde{p}_t\| + \beta \sum_{t=1}^T \lambda_t + \sum_{t=1}^T \eta_t. \end{aligned} \tag{8}$$

Noticing that  $h_t(0) = -\frac{B}{T} - \eta_t$ , take  $x = 0$  in formula (6), we obtain

$$-\sum_{t=1}^T F_t(x_t) \leq -\frac{B}{T} \sum_{t=1}^T \lambda_t + \frac{d^2}{2\alpha} + \alpha\delta^2 T + \frac{Ld^2}{2K} T.$$

Substituting the specific parameter values into it, one can deduce that

$$\beta \sum_{t=1}^T \lambda_t \leq \frac{\sqrt{2}d\delta^2}{2B} T \sqrt{T} + \frac{\sqrt{2}L\delta d^3}{4B} T + \frac{\delta^2 d^2}{B} T.$$

Thus, plugging the above formula back into formula (8), we get the conclusion by Lemma 4 and inequality  $\sum_{t \in [T]} \frac{1}{\sqrt{t}} \leq 2\sqrt{T+1}$ .

## 5 Conclusion

In this paper, we consider online optimization problems with stochastic long-term constraints, where the objective function is neither DR-submodular nor concave. Taking gradient ascent method as a subroutine, an algorithm of primal-dual type is proposed to solve it. We also analyze the performance of the proposed algorithm.

**Acknowledgements.** The second author is supported by Natural Science Foundation of China (No. 12101587) and China Postdoctoral Science Foundation (No. 2021M703167) and Fundamental Research Funds for the Central Universities (No. EIE40108X2).

## References

1. Balseiro, S.R., Gur, Y.: Learning in repeated auctions with budgets: regret minimization and equilibrium. *Manage. Sci.* **65**(9), 3952–3968 (2019)
2. Ferreira, K.J., Simchi-Levi, D., Wang, H.: Online network revenue management using thompson sampling. *Oper. Res.* **66**(6), 1586–1602 (2018)
3. Zinkevich, M.: Online convex programming and generalized infinitesimal gradient ascent. In: *Proceedings of the 20th International Conference on Machine Learning (icml-03)*, pp. 928–936 (2003)
4. Hazan, E.: Introduction to online convex optimization. *Found. Trends Optim.* **2**(3–4), 157–325 (2016)
5. Hazan, E., Agarwal, A., Kale, S.: Logarithmic regret algorithms for online convex optimization. *Mach. Learn.* **69**(2), 169–192 (2007)
6. Mannor, S., Tsitsiklis, J.N., Yu, J.Y.: Online learning with sample path constraints. *J. Mach. Learn. Res.* **10**(3), 569–590 (2009)
7. Neely, M.J., Yu, H.: Online convex optimization with time-varying constraints. arXiv preprint [arXiv:1702.04783](https://arxiv.org/abs/1702.04783) (2017)
8. Liakopoulos, N., Destounis, A., Paschos, G., et al.: Cautious regret minimization: online optimization with long-term budget constraints. In: *Proceedings of the 36th International Conference on Machine Learning*, pp. 3944–3952 (2019)
9. Sun, W., Dey, D., Kapoor, A.: Safety-aware algorithms for adversarial contextual bandit. In: *International Conference on Machine Learning*, pp. 3280–3288 (2017)
10. Bian, A.A., Mirzasoleiman, B., Buhmann, J., et al.: Guaranteed non-convex optimization: submodular maximization over continuous domains. In: *Artificial Intelligence and Statistics*, pp. 111–120 (2017)
11. Chen, L., Hassani, H., Karbasi, A.: Online continuous submodular maximization. In: *International Conference on Artificial Intelligence and Statistics*, pp. 1896–1905 (2018)
12. Chen, L., Harshaw, C., Hassani, H., et al.: Projection-free online optimization with stochastic gradient: from convexity to submodularity. In: *International Conference on Machine Learning*, pp. 814–823 (2018)
13. Zhang, M., Chen, L., Hassani, H., et al.: Online continuous submodular maximization: from full-information to bandit feedback. In: *Advances in Neural Information Processing Systems*, pp. 9210–9221 (2019)

14. Bian, A., Levy, K.Y., Krause, A., et al.: Non-monotone continuous DR-submodular maximization: structure and algorithms. *Adv. Neural. Inf. Process. Syst.* **30**, 487–497 (2018)
15. Dürr, C., Thắng, N.K., Srivastav, A., et al.: Non-monotone DR-submodular maximization: approximation and regret guarantees. In: *Proceedings of 29th International Joint Conference on Artificial Intelligence* (2020)
16. Thắng, N.K., Srivastav, A.: Online non-monotone DR-submodular maximization. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(11), pp. 9868–9876 (2021)
17. Sadeghi, O., Raut, P., Fazel, M.: A single recipe for online submodular maximization with adversarial or stochastic constraints. In: *Advances in Neural Information Processing Systems* 33, pp. 14712–14723 (2020)
18. Sadeghi, O., Fazel, M.: Online continuous DR-submodular maximization with long-term budget constraints. In: *International Conference on Artificial Intelligence and Statistics*, pp. 4410–4419 (2020)
19. Raut, P.S., Sadeghi, O., Fazel, M.: Online DR-submodular maximization with stochastic cumulative constraints. *arXiv preprint [arXiv:2005.14708](https://arxiv.org/abs/2005.14708)* (2020)
20. Hassani, H., Soltanolkotabi, M., Karbasi, A.: Gradient methods for submodular maximization. In: *Advances in Neural Information Processing Systems*, 30 (2017)
21. Jin, C., Netrapalli, P., Ge, R., et al.: A short note on concentration inequalities for random vectors with subgaussian norm. *arXiv preprint [arXiv:1902.03736](https://arxiv.org/abs/1902.03736)* (2019)



# Physical ZKP for Makaro Using a Standard Deck of Cards

Suthee Ruangwises<sup>(✉)</sup>  and Toshiya Itoh 

Department of Mathematical and Computing Science,  
Tokyo Institute of Technology, Tokyo, Japan  
ruangwises@gmail.com, titoh@c.titech.ac.jp

**Abstract.** Makaro is a logic puzzle with an objective to fill numbers into a rectangular grid to satisfy certain conditions. In 2018, Bultel et al. developed a physical zero-knowledge proof (ZKP) protocol for Makaro using a deck of cards, which allows a prover to physically convince a verifier that he/she knows a solution of the puzzle without revealing it. However, their protocol requires several identical copies of some cards, making it impractical as a deck of playing cards found in everyday life typically consists of all different cards. In this paper, we propose a new ZKP protocol for Makaro that can be implemented using a standard deck (a deck consisting of all different cards). Our protocol also uses asymptotically less cards than the protocol of Bultel et al. Most importantly, we develop a general method to encode a number with a sequence of all different cards. This allows us to securely compute several numerical functions using a standard deck, such as verifying that two given numbers are different and verifying that a number is the largest one among the given numbers.

**Keywords:** Zero-knowledge proof · Card-based cryptography · Makaro · Puzzle

## 1 Introduction

*Makaro* is a logic puzzle created by Nikoli, a company that developed many famous logic puzzles including Sudoku and Kakuro. A Makaro puzzle consists of a rectangular grid of white and black cells. White cells are divided into polyominoes called *rooms*, with some cells already containing a number, while each black cell contain an arrow pointing to some direction. The objective of this puzzle is to fill a number into each empty white cell according to the following rules [16].

1. *Room condition:* Each room must contain consecutive numbers starting from 1 to its *size* (the number of cells in the room).
2. *Neighbor condition:* Two (horizontally or vertically) adjacent cells in different rooms must contain different numbers.
3. *Arrow condition:* Each arrow in a black cell must point to the only largest number among the (up to) four numbers in the white cells adjacent to that black cell. See Fig. 1.

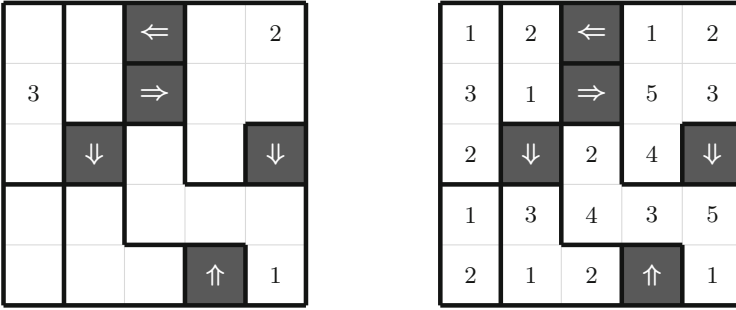


Fig. 1. An example of a Makaro puzzle (left) and its solution (right)

Determining whether a given Makaro puzzle has a solution has been proved to be NP-complete [7].

Suppose that Amber created a difficult Makaro puzzle and challenged her friend Bennett to solve it. After a while, Bennett could not solve her puzzle and began to doubt whether the puzzle has a solution. Amber needs to convince him that her puzzle actually has a solution without revealing it to him. In this situation, Amber needs a *zero-knowledge proof (ZKP)*.

### 1.1 Zero-Knowledge Proof

The concept of a ZKP was first introduced by Goldwasser et al. [4]. A ZKP is an interactive proof between  $P$  and  $V$  where both of them are given a computational problem  $x$ , but only  $P$  knows a solution  $w$  of  $x$ . A ZKP with perfect completeness and perfect soundness must satisfy the following three properties.

1. **Perfect Completeness:** If  $P$  knows  $w$ , then  $V$  always accepts.
2. **Perfect Soundness:** If  $P$  does not know  $w$ , then  $V$  always rejects.
3. **Zero-knowledge:**  $V$  learns nothing about  $w$ . Formally, there exists a probabilistic polynomial time algorithm  $S$  (called a *simulator*) that does not know  $w$  but has access to  $V$ , and the outputs of  $S$  follow the same probability distribution as the ones from the actual protocol.

Many recent results have been focusing on constructing physical ZKPs using objects found in everyday life such as a deck of cards and envelopes. These physical protocols have benefits that they do not require computers and also allow external observers to verify that the prover truthfully executes the protocol (which is often a challenging task for digital protocols). They are also suitable for teaching purpose and can be used to teach the concept of a ZKP to non-experts.

### 1.2 Related Work

**Protocol of Bultel et al.** In 2018, Bultel et al. [2] developed the first card-based ZKP protocol for Makaro. Their protocol uses  $\Theta(nk)$  cards, where  $n$  and



$k$  are the number of white cells and the size of the largest room, respectively. However, it requires  $\Theta(nk)$  identical copies of a specific card (and also  $\Theta(n)$  identical copies of another card).

As a deck of playing cards found in everyday life typically consists of all different cards,  $\Theta(nk)$  identical decks are actually required to implement this protocol, making the protocol very impractical. Another option is to use a different kind of deck (e.g. cards from board games) that contains several identical copies of some cards, but these decks are more difficult to find in everyday life.

**Other Protocols.** Besides Makaro, card-based ZKP protocols for many other logic puzzles have also been developed: Sudoku [5, 25], Akari [1], Takuzu [1, 12], Kakuro [1, 13], KenKen [1], Norinori [3], Slitherlink [10], Juosan [12], Numberlink [22], Suguru [18], Ripple Effect [23], Nurikabe [17], Hitori [17], Bridges [24], Masyu [10], Nonogram [19], Heyawake [17], and Shikaku [21]. All of these protocols, however, require a deck with repeated cards.

An open problem to develop ZKP protocols for logic puzzles using a standard deck (a deck consisting of all different cards) was posed by Koyama et al. [9]. This problem was recently answered by Ruangwises [20], who developed a ZKP protocol for Sudoku using a standard deck, the first standard deck protocol for any kind of logic puzzle. However, the protocol in [20] was specifically designed to tackle only the rules of Sudoku and cannot be applied to verify other numerical functions or other logic puzzles, thus having limited utility.

Other than logic puzzles, card-based protocols have also been widely studied in secure multi-party computation, a setting where multiple parties want to jointly compute a function of their secret inputs without revealing them. Almost all of existing protocols, however, also use a deck with repeated cards. The only exceptions are [8, 9, 11, 14, 15] which proposed AND, XOR, copy, and Yao’s millionaire protocols using a standard deck.

### 1.3 Our Contribution

Considering the drawback of the protocol of Bultel et al. [2], we aim to develop a more practical ZKP protocol for Makaro that can be implemented using a standard deck.<sup>1</sup>

In this paper, we propose a new ZKP protocol for Makaro with perfect completeness and soundness using a standard deck. It is also the second standard deck protocol for any logic puzzle, after the one for Sudoku [20]. Remarkably, our protocol uses asymptotically less cards than the protocol of Bultel et al. (see Table 1). This is a noteworthy achievement as card-based protocols that use a standard deck generally require more cards than their counterparts that use a deck with repeated cards [26]. (In particular, the standard deck protocol for Sudoku [20] also requires more cards than its counterpart [25].)

---

<sup>1</sup> Although a “standard deck” of playing cards found in everyday life typically consists of 52 different cards, in theory we study a general setting where the deck is arbitrarily large, consisting of all different cards.

**Table 1.** The number of required cards for each protocol for Makaro, where  $n$  and  $k$  are the number of white cells and the size of the largest room, respectively

Protocol	Standard Deck?	#Cards
Bultel et al. [2]	No	$\Theta(nk)$
Ours	Yes	$\Theta(n + k)$

Most importantly, we develop a general method to encode a number with a sequence of all different cards. This allows us to securely compute several numerical functions using a standard deck, such as verifying that two given numbers are different and verifying that a number is the largest one among the given numbers.

## 2 Preliminaries

Let  $n$  be the number of white cells and  $k$  be the size of the largest room in the Makaro grid.

We assume that all cards used in our protocols have different front sides and identical back sides. For didactic purpose, cards are divided into *sets*. Cards in the same set are denoted by the same letter with different indices, e.g. cards  $a_1, a_2, a_3, a_4$  are in the same set.

In an  $\ell \times m$  *matrix* of cards, let Row  $i$  denote the  $i$ -th topmost row, and Column  $j$  denote the  $j$ -th leftmost column.

### 2.1 Pile-Shifting Shuffle

Given an  $\ell \times m$  matrix of cards, a *pile-shifting shuffle* [27] rearranges the columns of the matrix by a random cyclic shift unknown to all parties. It can be implemented in real world by putting the cards in each column into an envelope and then taking turns to apply *Hindu cuts* (taking several envelopes from the bottom and putting them on the top) to the pile of envelopes [28].

### 2.2 Pile-Scramble Shuffle

Given an  $\ell \times m$  matrix of cards, a *pile-scramble shuffle* [6] rearranges the columns of the matrix by a random permutation unknown to all parties. It can be implemented in real world by putting the cards in each column into an envelope and then jointly scrambling the envelopes together randomly.

## 3 Main Protocol

### 3.1 Cell Cards

We use a *cell card* to represent each white cell in the grid. Cells in the same room are represented by cards in the same set. To avoid confusion, a cell card is

always denoted by a Greek letter followed by an index equal to the number in the cell it represents. We have cell cards in sets  $\alpha_i, \beta_i, \gamma_i, \dots$  and so on. See Fig. 2 for an example.<sup>2</sup>

$\alpha_1$	$\beta_2$	$\Leftarrow$	$\gamma_1$	$\gamma_2$
$\alpha_3$	$\beta_1$	$\Rightarrow$	$\gamma_5$	$\gamma_3$
$\alpha_2$	$\Downarrow$	$\delta_2$	$\gamma_4$	$\Downarrow$
$\epsilon_1$	$\zeta_3$	$\delta_4$	$\delta_3$	$\delta_5$
$\epsilon_2$	$\zeta_1$	$\zeta_2$	$\Uparrow$	$\delta_1$

**Fig. 2.** A cell card representing each white cell in the solution of the puzzle in Fig. 1

At the beginning,  $P$  publicly places a face-down corresponding cell card on each white cell already having a number. Then,  $P$  secretly places a face-down corresponding cell card according to his/her solution on each empty white cell.

### 3.2 Verifying Room Condition

Consider a room  $R$  of size  $p$  in the Makaro grid containing cells represented by cell cards  $\alpha_1, \alpha_2, \dots, \alpha_p$ . This subprotocol allows  $P$  to show that the cell cards in  $R$  consist of a permutation of  $\alpha_1, \alpha_2, \dots, \alpha_p$  without revealing their order. It was developed by Sasaki et al. [25].

Besides cell cards, we also use *helping cards*  $h_i$  ( $i = 1, 2, \dots, k$ ) in our protocol.

$?$	$?$	$\dots$	$?$
$\alpha_?$	$\alpha_?$		$\alpha_?$
$?$	$?$	$\dots$	$?$
$h_1$	$h_2$		$h_p$

**Fig. 3.** A  $2 \times p$  matrix constructed in Step 2

---

<sup>2</sup> Assume that we have  $\ell$  cards with different numbers, e.g. cards with numbers  $1, 2, \dots, \ell$ . In the example in Fig. 2, we can, for instance, regard cards 1, 2, 3 on cells with numbers 1, 2, 3 in the top-left room as  $\alpha_1, \alpha_2, \alpha_3$ , cards 4, 5 on cells with numbers 1, 2 in the top-center room as  $\beta_1, \beta_2$ , cards 6, 7, 8, 9, 10 on cells with numbers 1, 2, 3, 4, 5 in the top-right room as  $\gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5$ , and so on.

1. Take all cell cards in  $R$  in any specific order (e.g. from top to bottom, then from left to right) and place them face-down in Row 1 of a matrix  $M$ .
2. Publicly place face-down helping cards  $h_1, h_2, \dots, h_p$  in Row 2 of  $M$  in this order from left to right.  $M$  is now a  $2 \times p$  matrix (see Fig. 3).
3. Apply the pile-scramble shuffle to  $M$ .
4. Turn over all cards in Row 1 of  $M$ . If the sequence is a permutation of  $\alpha_1, \alpha_2, \dots, \alpha_p$ , proceed to the next step; otherwise,  $V$  rejects.
5. Turn over all face-up cards in  $M$ . Apply the pile-scramble shuffle to  $M$  again.
6. Turn over all cards in Row 2 of  $M$ . Arrange the columns of  $M$  such that the cards in Row 2 are  $h_1, h_2, \dots, h_p$  in this order from left to right. Note that the columns of  $M$  are now reverted to their original order.
7. Take the cards in Row 1 of  $M$  and place them back into room  $R$  in the same order we take them in Step 1.

$P$  applies this subprotocol for every room in the Makaro grid to verify the room condition.

However, verifying the neighbor condition and arrow condition is more difficult and cannot be done by using cell cards alone. Therefore, we have to develop a method to encode a number with a sequence of all different cards.

### 3.3 Encoding Sequences

In previous ZKP protocols for other logic puzzles [2, 18, 22–24], a number  $x$  ( $1 \leq x \leq m$ ) is often encoded by a sequence  $E_m(x)$  of  $m$  consecutive cards, with all of them being  $\clubsuit$ s except the  $x$ -th leftmost card being a  $\heartsuit$  (e.g.  $E_4(2)$  is  $\clubsuit \heartsuit \clubsuit \clubsuit$ ). We will employ that idea to develop an encoding sequence for a number  $x$  using all different cards.

Besides cell cards and helping cards, we also use *encoding cards*  $a_i, b_i, c_i, d_i$  ( $i = 1, 2, \dots, 2k - 1$ ) in our protocol. (We need four sets of encoding cards because we later have to compare up to four numbers at the same time during the arrow condition verification.)

For a fixed integer  $m \leq 2k - 1$ , define a sequence  $E_m^a(x)$  to be a sequence of  $m$  consecutive cards, where the  $x$ -th leftmost card is  $a_1$ , and the other  $m - 1$  cards are a uniformly random permutation of  $a_2, a_3, \dots, a_m$  unknown to  $V$ .

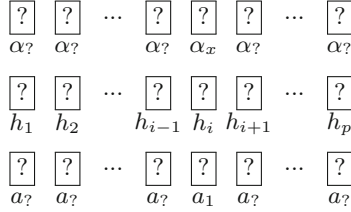
The role of the card  $a_1$  in  $E_m^a(x)$  is to mark the value of  $x$ , similarly to a  $\heartsuit$  in  $E_m(x)$ . Note that the order of  $a_2, a_3, \dots, a_m$  must be unknown to  $V$  in order for the protocol to be zero-knowledge, so each encoding sequence is for one-time use only.

We also define sequences  $E_m^b(x)$ ,  $E_m^c(x)$ , and  $E_m^d(x)$  analogously, using encoding cards from sets  $b_i, c_i$ , and  $d_i$ , with cards  $b_1, c_1$ , and  $d_1$  as marking points, respectively.

### 3.4 Conversion from Cell Cards to Encoding Sequences

This is the most crucial subprotocol in our protocol. Let  $w$  be any white cell represented by a cell card  $\alpha_x$ . Suppose that  $w$  is located in a room  $R$  with size

$p$ . This subprotocol allows  $P$  to construct an encoding sequence  $E_m^a(x)$  for some fixed  $m \geq p$  without revealing the value  $x$  to  $V$ , while leaving all cell cards in  $R$  unchanged.



**Fig. 4.** A  $3 \times k$  matrix  $M$  constructed in Step 5

1. Take all cell cards in  $R$  in any specific order (e.g. from top to bottom, then from left to right) and place them face-down in Row 1 of a matrix  $M$ . Suppose the card  $\alpha_x$  is located at Column  $i$  of  $M$ .
2. Publicly place face-down helping cards  $h_1, h_2, \dots, h_p$  in Row 2 of  $M$  in this order from left to right.
3. Publicly place face-down encoding card  $a_1$  in Row 3 of  $M$  at Column  $i$ .
4. Secretly arrange face-down encoding cards  $a_2, a_3, \dots, a_m$  in a uniformly random permutation unknown to  $V$ . Refer to this sequence as  $S$ .
5. Take the  $p - 1$  leftmost cards of  $S$  and place them in empty cells in Row 3 of  $M$  in this order from left to right. Leave the  $m - p$  rightmost cards of  $S$  unchanged.  $M$  is now a complete  $3 \times p$  matrix (see Fig. 4).
6. Apply the pile-scramble shuffle to  $M$ .
7. Turn over all cards in Row 1 of  $M$ . Arrange the columns of  $M$  such that the cards in Row 1 are  $\alpha_1, \alpha_2, \dots, \alpha_p$  in this order from left to right.
8. Take all cards in Row 3 of  $M$  out of the matrix ( $M$  now becomes a  $2 \times p$  matrix). Refer to the sequence taken from Row 3 of  $M$  as  $T$ . Append the  $m - p$  rightmost cards of  $S$  left in Step 5 to the right of  $T$ . The appended sequence is  $E_m^a(x)$  as desired.
9. Turn over all face-up cards in  $M$ . Apply the pile-scramble shuffle to  $M$  again.
10. Turn over all cards in Row 2 of  $M$ . Arrange the columns of  $M$  such that the cards in Row 2 are  $h_1, h_2, \dots, h_p$  in this order from left to right. Note that the columns of  $M$  are now reverted to their original order.
11. Take the cards in Row 1 of  $M$  and place them back into room  $R$  in the same order we take them in Step 1.

### 3.5 Verifying Neighbor Condition

This subprotocol allows  $P$  to show that two adjacent cells represented by  $\alpha_x$  and  $\beta_y$  in different rooms contain different numbers. The idea of this subprotocol is

exactly the same as the one developed by Bultel et al. [2, §3.3 Step 2] to verify the same condition, except that it uses encoding sequences  $E_m^a(x)$  and  $E_m^b(y)$  instead of  $E_m(x)$  and  $E_m(y)$ .

Let  $p$  and  $q$  be the sizes of rooms containing  $\alpha_x$  and  $\beta_y$ , respectively, and let  $m = \max(p, q)$ . Note that we have  $m \leq k$ . First,  $P$  applies the conversion protocol in Sect. 3.4 to construct sequences  $E_m^a(x)$  and  $E_m^b(y)$  from  $\alpha_x$  and  $\beta_y$ , respectively. Then,  $P$  performs the following steps.

1. Construct a  $2 \times m$  matrix  $M$  by placing  $E_m^a(x)$  and  $E_m^b(y)$  in Row 1 and Row 2, respectively.
2. Apply the pile-scramble shuffle to  $M$ .
3. Turn over all cards in Row 1 of  $M$ . Suppose  $a_1$  is located at Column  $i$ .
4. Turn over a card in Row 2 of  $M$  at Column  $i$ . If it is not  $b_1$ , proceed to the next step; otherwise,  $V$  rejects.

$P$  applies this subprotocol for every pair of adjacent cells that are in different rooms in the Makaro grid to verify the neighbor condition.

### 3.6 Verifying Arrow Condition

Suppose the (up to) four cells adjacent to a black cell containing an arrow are represented by  $\alpha_x$ ,  $\beta_y$ ,  $\gamma_z$ , and  $\delta_t$ , with an arrow pointing to  $\alpha_x$ .<sup>3</sup> This subprotocol allows  $P$  to show that a number in the cell represented by  $\alpha_x$  is the largest one among all numbers in these cells. The idea of this subprotocol is exactly the same as the one developed by Bultel et al. [2, §3.3 Step 3] to verify the same condition, except that it uses encoding sequences  $E_{2m-1}^a(x)$ ,  $E_{2m-1}^b(y)$ ,  $E_{2m-1}^c(z)$ , and  $E_{2m-1}^d(t)$  instead of  $E_{2m-1}(x)$ ,  $E_{2m-1}(y)$ ,  $E_{2m-1}(z)$ , and  $E_{2m-1}(t)$ .

Let  $p$ ,  $q$ ,  $r$ , and  $s$  be the sizes of rooms containing  $\alpha_x$ ,  $\beta_y$ ,  $\gamma_z$ , and  $\delta_t$ , respectively, and let  $m = \max(p, q, r, s)$ . Note that we have  $m \leq k$ , and thus  $2m - 1 \leq 2k - 1$ . First,  $P$  applies the conversion protocol in Sect. 3.4 to construct sequences  $E_{2m-1}^a(x)$ ,  $E_{2m-1}^b(y)$ ,  $E_{2m-1}^c(z)$ , and  $E_{2m-1}^d(t)$  from  $\alpha_x$ ,  $\beta_y$ ,  $\gamma_z$ , and  $\delta_t$ , respectively. Then,  $P$  performs the following steps.

1. Construct a  $4 \times (2m-1)$  matrix  $M$  by placing  $E_{2m-1}^a(x)$ ,  $E_{2m-1}^b(y)$ ,  $E_{2m-1}^c(z)$ , and  $E_{2m-1}^d(t)$  in Rows 1, 2, 3, and 4, respectively.
2. Apply the pile-shifting shuffle to  $M$ .
3. Turn over all cards in Row 1 of  $M$ . Suppose  $a_1$  is located at Column  $i$ .
4. Turn over cards in Rows 2, 3, and 4 of  $M$  at Columns  $i$ ,  $i + 1$ , ...,  $i + m - 1$  (where the indices are taken modulo  $2m - 1$ ). If none of them is  $b_1$ ,  $c_1$ , or  $d_1$ , proceed to the next step; otherwise,  $V$  rejects.

$P$  applies this subprotocol for every arrow in the Makaro grid to verify the arrow condition.

If the verification passes for all three conditions, then  $V$  accepts.

---

<sup>3</sup> Some of the cells may be in the same room, but this does not affect the conversion as we apply the conversion protocol to each cell card one by one.

### 3.7 Complexity

Our protocol uses  $n$  cell cards,  $k$  helping cards, and  $4(2k - 1)$  encoding cards, resulting in the total of  $n + 9k - 4 = \Theta(n + k)$  cards. In comparison, the protocol of Bultel et al. [2] requires  $\Theta(nk)$  cards.

## 4 Proof of Correctness and Security

We will prove the perfect completeness, perfect soundness, and zero-knowledge properties of our protocol.

**Lemma 1 (Perfect Completeness).** *If  $P$  knows a solution of the Makaro puzzle, then  $V$  always accepts.*

**Proof.** Suppose  $P$  knows a solution and places cards on the grid accordingly.

First, we will prove the correctness of the conversion protocol in Sect. 3.4. From the way we construct the matrix  $M$ , in Step 5 the card  $a_1$  is in the same column as  $\alpha_x$ , and the other  $p - 1$  cards in Row 3 are uniformly distributed among all  $\frac{(m-1)!}{(m-p)!}$  permutations of  $p - 1$  cards selected from  $a_2, a_3, \dots, a_m$ . In Step 7, the card  $a_1$  is moved to Column  $x$ . Hence, the appended sequence in Step 8 has  $a_1$  as the  $x$ -th leftmost card, and the other  $m - 1$  cards are uniformly distributed among all  $(m - 1)!$  permutations of  $a_2, a_3, \dots, a_m$  (which remains unknown to  $V$ ). Therefore, the appended sequence is indeed  $E_m^a(x)$ .

Next, we will prove that the verification of all three conditions will pass.

- For the room condition verification in Sect. 3.2, the cards that are turned over in Step 4 must be a permutation of  $\alpha_1, \alpha_2, \dots, \alpha_p$ , so the verification will pass.
- For the neighbor condition verification in Sect. 3.5, the cell cards are correctly converted to sequences  $E_m^a(x)$  and  $E_m^b(y)$ . Since  $x \neq y$ , the cards  $a_1$  and  $b_1$  must be in different columns of  $M$ . Hence, the card that is turned over in Step 4 cannot be  $b_1$ , so the verification will pass.
- For the arrow condition verification in Sect. 3.6, the cell cards are correctly converted to sequences  $E_{2m-1}^a(x)$ ,  $E_{2m-1}^b(y)$ ,  $E_{2m-1}^c(z)$ , and  $E_{2m-1}^d(t)$ . Since  $x$  is the only largest number among the four numbers, in Step 3 each of the cards  $b_1$ ,  $c_1$ , and  $d_1$  must be in one of Columns  $i - 1, i - 2, \dots, i - m + 1$  of  $M$  (where the indices are taken modulo  $2m - 1$ ). Hence, the cards that are turned over in Step 4 cannot include  $b_1$ ,  $c_1$ , or  $d_1$ , so the verification will pass.

Therefore,  $V$  always accepts.  $\square$

**Lemma 2 (Perfect Soundness).** *If  $P$  does not know a solution of the Makaro puzzle, then  $V$  always rejects.*

**Proof.** Suppose  $P$  does not know a solution. At least one of the three conditions must be violated.

- If the room condition is violated, consider the room condition verification in Sect. 3.2 for a room that violates the condition. The cards that are turned over in Step 4 cannot be a permutation of  $\alpha_1, \alpha_2, \dots, \alpha_p$ , so the verification will fail.
- If the neighbor condition is violated, consider the neighbor condition verification in Sect. 3.5 for a pair of adjacent cells that violates the condition. We have  $x = y$ , so the cards  $a_1$  and  $b_1$  must be in the same column of  $M$ . Hence, the card that is turned over in Step 4 must be  $b_1$ , so the verification will fail.
- If the arrow condition is violated, consider the arrow condition verification in Sect. 3.6 for an arrow that violates the condition. Suppose  $y \geq x$ . In Step 3, the card  $b_1$  must be in one of Columns  $i, i + 1, \dots, i + m - 1$  of  $M$  (where the indices are taken modulo  $2m - 1$ ). Hence, the cards that are turned over in Step 4 must include  $b_1$ , so the verification will fail.

Therefore,  $V$  always rejects. □

**Lemma 3 (Zero-Knowledge).** *During the verification,  $V$  learns nothing about  $P$ 's solution.*

**Proof.** It is sufficient to show that all distributions of cards that are turned face-up can be simulated by a simulator  $S$  that does not know  $P$ 's solution.

- In the room condition verification in Sect. 3.2:
  - In Step 4, the orders of face-up cards are uniformly distributed among all  $p!$  permutations of  $\alpha_1, \alpha_2, \dots, \alpha_p$ , so it can be simulated by  $S$ .
  - In Step 6, the orders of face-up cards are uniformly distributed among all  $p!$  permutations of  $h_1, h_2, \dots, h_p$ , so it can be simulated by  $S$ .
- In the conversion protocol in Sect. 3.4:
  - In Step 7, the orders of face-up cards are uniformly distributed among all  $p!$  permutations of  $\alpha_1, \alpha_2, \dots, \alpha_p$ , so it can be simulated by  $S$ .
  - In Step 10, the orders of face-up cards are uniformly distributed among all  $p!$  permutations of  $h_1, h_2, \dots, h_p$ , so it can be simulated by  $S$ .
- In the neighbor condition verification in Sect. 3.5:
  - In Step 3, the orders of face-up cards are uniformly distributed among all  $m!$  permutations of  $a_1, a_2, \dots, a_m$ , so it can be simulated by  $S$ .
  - In Step 4, the face-up card has an equal probability to be one of  $b_2, b_3, \dots, b_m$ , so it can be simulated by  $S$ .
- In the arrow condition verification in Sect. 3.6:
  - In Step 3, the orders of face-up cards are uniformly distributed among all  $(2m - 1)!$  permutations of  $a_1, a_2, \dots, a_{2m-1}$ , so it can be simulated by  $S$ .
  - In Step 4, the orders of face-up cards in Row 2 are uniformly distributed among all  $\frac{(2m-2)!}{(m-2)!}$  permutations of  $m$  cards selected from  $b_2, b_3, \dots, b_{2m-1}$ . The same goes for face-up cards in Row 3 and Row 4, with  $m$  cards selected from  $c_2, c_3, \dots, c_{2m-1}$  and  $d_2, d_3, \dots, d_{2m-1}$ , respectively. So, it can be simulated by  $S$ .

Therefore, we can conclude that  $V$  learns nothing about  $P$ 's collusion. □



## 5 Future Work

We developed a ZKP protocol for Makaro using a standard deck, which requires asymptotically less cards than the existing protocol of Bultel et al. [2]. We also developed a general method to encode a number with a sequence of all different cards, which allows us to securely compute several numerical functions using a standard deck. This method can be used to verify solutions of some other logic puzzles including Suguru. Possible future work includes developing standard deck protocols to verify solutions of other logic puzzles (e.g. Kakuro, Numberlink), or to compute broader types of functions.



## References

1. Bultel, X., Dreier, J., Dumas, J.-G., Lafourcade, P.: Physical zero-knowledge proofs for Akari, Takuzu, Kakuro and KenKen. In: Proceedings of the 8th International Conference on Fun with Algorithms (FUN), pp. 8:1–8:20 (2016)
2. Bultel, X., et al.: Physical zero-knowledge proof for Makaro. In: Proceedings of the 20th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS), pp. 111–125 (2018)
3. Dumas, J.-G., Lafourcade, P., Miyahara, D., Mizuki, T., Sasaki, T., Sone, H.: Interactive physical zero-knowledge proof for Norinori. In: Proceedings of the 25th International Computing and Combinatorics Conference (COCOON), pp. 166–177 (2019)
4. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. *SIAM J. Comput.* **18**(1), 186–208 (1989)
5. Gradwohl, R., Naor, M., Pinkas, B., Rothblum, G.N.: Cryptographic and physical zero-knowledge proof systems for solutions of sudoku puzzles. *Theory Comput. Syst.* **44**(2), 245–268 (2009)
6. Ishikawa, R., Chida, E., Mizuki, T.: Efficient card-based protocols for generating a hidden random permutation without fixed points. In: Proceedings of the 14th International Conference on Unconventional Computation and Natural Computation (UCNC), pp. 215–226 (2015)
7. Iwamoto, C., Haruishi, M., Ibusuki, T.: Herugolf and Makaro are NP-complete. In: Proceedings of the 9th International Conference on Fun with Algorithms (FUN), pp. 24:1–24:11 (2018)
8. Koch, A., Schrempf, M., Kirsten, M.: Card-based cryptography meets formal verification. *N. Gener. Comput.* **39**(1), 115–158 (2021)
9. Koyama, H., Miyahara, D., Mizuki, T., Sone, H.: A secure three-input AND protocol with a standard deck of minimal cards. In: Proceedings of the 16th International Computer Science Symposium in Russia (CSR), pp. 242–256 (2021)
10. Lafourcade, P., Miyahara, D., Mizuki, T., Robert, L., Sasaki, T., Sone, H.: How to construct physical zero-knowledge proofs for puzzles with a “single loop” condition. *Theoret. Comput. Sci.* **888**, 41–55 (2021)
11. Miyahara, D., Hayashi, Y., Mizuki, T., Sone, H.: Practical card-based implementations of Yao’s millionaire protocol. *Theoret. Comput. Sci.* **803**, 207–221 (2020)
12. Miyahara, D., et al.: Card-based ZKP protocols for Takuzu and Juosan. In: Proceedings of the 10th International Conference on Fun with Algorithms (FUN), pp. 20:1–20:21 (2020)

13. Miyahara, D., Sasaki, T., Mizuki, T., Sone, H.: Card-based physical zero-knowledge proof for kakuro. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **E102.A(9)**, 1072–1078 (2019)
14. Mizuki, T.: Efficient and secure multiparty computations using a standard deck of playing cards. In: *Proceedings of the 15th International Conference on Cryptology and Network Security (CANS)*, pp. 484–499 (2016)
15. Niemi, V., Renvall, A.: Solitaire zero-knowledge. *Fundamenta Informaticae* **38(1,2)**, 181–188 (1999)
16. Nikoli: Makaro. <https://www.nikoli.co.jp/en/puzzles/makaro/>
17. Robert, L., Miyahara, D., Lafourcade, P., Mizuki, T.: Card-based ZKP for connectivity: applications to Nurikabe, Hitori, and Heyawake. *N. Gener. Comput.* **40(1)**, 149–171 (2022)
18. Robert, L., Miyahara, D., Lafourcade, P., Libralesso, L., Mizuki, T.: Physical zero-knowledge proof and NP-completeness proof of Suguru puzzle. *Inf. Comput.* **285(B)**, 104858 (2022)
19. Ruangwises, S.: An improved physical ZKP for nonogram. In: *Proceedings of the 15th Annual International Conference on Combinatorial Optimization and Applications (COCOA)*, pp. 262–272 (2021)
20. Ruangwises, S.: Two standard decks of playing cards are sufficient for a ZKP for sudoku. *N. Gener. Comput.* **40(1)**, 49–65 (2022)
21. Ruangwises, S., Itoh, T.: How to physically verify a rectangle in a grid: a physical ZKP for Shikaku. In: *Proceedings of the 11th International Conference on Fun with Algorithms (FUN)*, pp. 24:1–24:12 (2022)
22. Ruangwises, S., Itoh, T.: Physical zero-knowledge proof for numberlink puzzle and  $k$  vertex-disjoint paths problem. *N. Gener. Comput.* **39(1)**, 3–17 (2021)
23. Ruangwises, S., Itoh, T.: Physical zero-knowledge proof for ripple effect. *Theoret. Comput. Sci.* **895**, 115–123 (2021)
24. Ruangwises, S., Itoh, T.: Physical ZKP for connected spanning subgraph: applications to bridges puzzle and other problems. In: *Proceedings of the 19th International Conference on Unconventional Computation and Natural Computation (UCNC)*, pp. 149–163 (2021)
25. Sasaki, T., Miyahara, D., Mizuki, T., Sone, H.: Efficient card-based zero-knowledge proof for Sudoku. *Theoret. Comput. Sci.* **839**, 135–142 (2020)
26. Shinagawa, K., Mizuki, T.: Secure computation of any boolean function based on any deck of cards. In: *Proceedings of the 13th International Frontiers of Algorithms Workshop (FAW)*, pp. 63–75 (2019)
27. Shinagawa, K., et al.: Card-based protocols using regular polygon cards. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **E100.A(9)**, 1900–1909 (2017)
28. Ueda, I., Miyahara, D., Nishimura, A., Hayashi, Y., Mizuki, T., Sone, H.: Secure implementations of a random bisection cut. *Int. J. Inf. Secur.* **19(4)**, 445–452 (2020)



# Characterization of the Imbalance Problem on Complete Bipartite Graphs

Steven Ge<sup>(✉)</sup>  and Toshiya Itoh 

Tokyo Institute of Technology, Meguro, Japan  
ge.s.aa@m.titech.ac.jp, titoh@c.titech.ac.jp

**Abstract.** We study the imbalance problem on complete bipartite graphs. The imbalance problem is a graph layout problem and is known to be NP-complete. Graph layout problems find their applications in the optimization of networks for parallel computer architectures, VLSI circuit design, information retrieval, numerical analysis, computational biology, graph theory, scheduling and archaeology [2]. In this paper, we give characterizations for the optimal solutions of the imbalance problem on complete bipartite graphs. Using the characterizations, we can solve the imbalance problem in time polylogarithmic in the number of vertices, when given the cardinalities of the parts of the graph, and verify whether a given solution is optimal in time linear in the number of vertices on complete bipartite graphs. We also introduce a generalized form of complete bipartite graphs on which the imbalance problem is solvable in time quasilinear in the number of vertices by using the aforementioned characterizations.

**Keywords:** Imbalance problem · Vertex layout · Complete bipartite graph · Proper interval bipartite graph

## 1 Introduction

Graph layout problems are combinatorial optimization problems, where the goal is to find an ordering on the vertices that optimizes an objective function. A large number of problems from different domains can be formulated as graph layout problems [2]. The imbalance problem is a graph layout problem that has applications in 3-dimensional circuit design [7].

The imbalance problem was introduced by Biedl et al. [1]. Given an ordering of the vertices of a graph  $G$ , the imbalance of a vertex  $v$  is the absolute difference in the number of neighbors to the left of  $v$  and the number of neighbors to the right of  $v$ . The imbalance of an ordering is the sum of the imbalances of the vertices. An instance of the imbalance problem consists of a graph  $G$  and an integer  $k$ . The problem asks whether there exists an ordering on the vertices of  $G$  such that the imbalance of the ordering is at most  $k$ .

The imbalance problem is NP-complete for several graph classes, including bipartite graphs with degree at most 6, weighted trees [1], general graphs with

degree at most 4 [6], and split graphs [4]. The problem becomes polynomial time solvable on superfragile graphs [4]. The problem is linear time solvable on proper interval graphs [4], bipartite permutation graphs, and threshold graphs [3].

Gorzny showed that the minimum imbalance of a bipartite permutation graph  $G = (V, E)$ , which is a superclass of complete bipartite graphs and proper interval bipartite graphs, can be computed in  $\mathcal{O}(|V| + |E|)$  time [3]. We give characterizations for the optimal solutions of the imbalance problem on complete bipartite graphs. Using the characterizations, we show that the imbalance problem is solvable in  $\mathcal{O}(\log(|V|) \cdot \log(\log(|V|)))$  time on complete bipartite graphs, when given the cardinalities of the parts of the graph. Additionally, using the characterizations, we can verify whether a given solution is optimal in  $\mathcal{O}(|V|)$  on complete bipartite graphs. We also introduce a generalized form of complete bipartite graphs, which we call chained complete bipartite graphs, on which the imbalance problem is solvable in  $\mathcal{O}(c \cdot \log(|V|) \cdot \log(\log(|V|)))$  time, where  $c = \mathcal{O}(|V|)$ , by using the aforementioned characterizations. As chained complete bipartite graphs are a subclass of proper interval bipartite graphs, the result of Gorzny also applies to chained complete bipartite graphs.

## 2 Preliminaries

We only consider graphs that are finite, undirected, connected, and simple (i.e. without multiple edges or loops). A graph  $G$  is denoted as  $G = (V, E)$ , where  $V$  denotes the set of vertices and  $E \subseteq V \times V$  denotes the set of undirected edges. For convenience, we abbreviate bipartite graph as bigraph.

For  $n \in \mathbb{N}^+$ , let us define  $[n] = \{1, 2, \dots, n\}$ .

We define  $\sigma_S : S \rightarrow [|S|]$  to be an ordering of  $S$ . For convenience, at times we denote an ordering  $\sigma_S$  by  $(\sigma^{-1}(1), \sigma^{-1}(2), \dots, \sigma^{-1}(|S|))$ , where,  $v = \sigma^{-1}(\sigma(v))$  for  $v \in S$ . We also say that  $v \in S$  is at position  $k$  in ordering  $\sigma_S$  if  $\sigma_S(v) = k$ .

Let  $S_1, S_2, \dots, S_n$  be a collection of disjoint sets. Let  $s \in S_i$ , where  $1 \leq i \leq n$ , then the concatenation of orderings is defined as follows:  $\sigma_{S_1} \sigma_{S_2} \dots \sigma_{S_n}(s) = \sigma_{S_i}(s) + \sum_{j \in [i-1]} |S_j|$ . Additionally, we use the product notation to denote the

concatenation over a set. That is,  $\prod_{i \in [n]} \sigma_{S_i} = \sigma_{S_1} \sigma_{S_2} \dots \sigma_{S_n}$ .

Given an ordering  $\sigma_S$  we say that  $v \in S$  occurs to the left of  $u \in S$  in  $\sigma_S$ , if and only if  $\sigma_S(v) < \sigma_S(u)$ . We denote this as  $v <_{\sigma_S} u$ . We define  $>_{\sigma_S}$  analogously.

Given an ordering  $\sigma_S$  we say that  $\sigma_S^{S'}$ , where  $S' \subseteq S$ , is a subordering of  $\sigma_S$  on  $S'$  if  $\sigma_S^{S'}$  preserves the relative ordering of the elements in  $S'$ . That is  $\sigma_S^{S'}$  is a subordering of  $\sigma_S$  if and only if  $\forall u, v \in S' \forall \oplus \in \{<, >\} u \oplus_{\sigma_S^{S'}} v \iff u \oplus_{\sigma_S} v$ .

For a graph  $G = (V, E)$ , the open neighborhood of a vertex  $v \in V$ , denoted by  $N(v)$ , is the set of vertices adjacent to  $v$ . We call the vertices in  $N(v)$  the neighbors of vertex  $v$ . That is  $N(v) = \{u \in V \mid \{u, v\} \in E\}$ .

The imbalance of a vertex  $v \in V$  on the ordering  $\sigma_V$  in graph  $G = (V, E)$ , denoted by  $I(v, \sigma_V, G)$ , is defined to be the absolute difference in the number

of neighbors of  $v$  occurring to the left of  $v$  and the number of neighbors of  $v$  occurring to the right of  $v$  in  $\sigma_V$ . That is,  $I(v, \sigma_V, G) = |\{u \in N(v) \mid u <_{\sigma_V} v\}| - |\{u \in N(v) \mid u >_{\sigma_V} v\}|$ .

The imbalance of an ordering  $\sigma_V$  on graph  $G = (V, E)$ , denoted by  $I(\sigma_V, G)$ , is defined to be sum over the imbalances of the vertices in  $V$  on the ordering  $\sigma_V$  in graph  $G$ . That is  $I(\sigma_V, G) = \sum_{v \in V} I(v, \sigma_V, G)$ . If the ordering and/or graph are clear from the context, we shall exclude them from the parameters of the function  $I$ .

The imbalance of a graph  $G = (V, E)$ , denoted by  $I(G)$ , is defined to be the minimum imbalance over all orderings  $\sigma_V$ . That is,  $I(G) = \min_{\sigma_V \in S(V)} I(\sigma_V, G)$ ,

where  $S(V)$  denotes the set of all orderings on  $V$ . We call an ordering  $\sigma_V$  whose imbalance is equivalent to  $I(G)$  an (imbalance) optimal ordering. Given a graph  $G$  and an integer  $k$ , the imbalance problem asks whether  $I(G) \leq k$  is true or false.

A bigraph  $G = (X, Y, E)$  is an interval bigraph, if there exists a set of intervals on the real line, where for each vertex  $v \in X \cup Y$  we have exactly one corresponding interval such that the intervals corresponding to vertices  $x \in X$  and  $y \in Y$  intersect if and only if there exists an edge in  $E$  connecting  $x$  and  $y$ . We call such a set of intervals the interval representation  $I_G$  of graph  $G$ . That is,  $I_G = \{[l_v, r_v] \mid v \in X \cup Y\}$  such that  $\forall_{x \in X} \forall_{y \in Y} ([l_x, r_x] \cap [l_y, r_y] \neq \emptyset \iff \{x, y\} \in E)$ .

An interval bigraph  $G = (X, Y, E)$  is a proper interval bigraph, abbreviated by PI-bigraph, if it has an interval representation  $I_G$  such that none of the intervals are properly contained in another. That is, there exists an interval representation  $I_G = \{[l_v, r_v] \mid v \in X \cup Y\}$  of  $G$  such that  $\forall_{u, v \in X \cup Y} (u \neq v \iff [l_u, r_u] \not\subseteq [l_v, r_v])$ .

### 3 Imbalance on Complete Bipartite Graphs

Let  $G = (X, Y, E)$  be a complete bigraph. In this section we shall prove that the minimum imbalance of  $G$  is  $|X| \cdot |Y| + (|X| \bmod 2) \cdot (|Y| \bmod 2)$ .

**Lemma 1.** *If  $G = (X, Y, E)$  is a complete bigraph, then there exists an ordering  $\sigma_{X \cup Y}$  such that  $I(\sigma_{X \cup Y}) = |X| \cdot |Y| + (|X| \bmod 2) \cdot (|Y| \bmod 2)$ .*

*Proof.* We shall construct an ordering on  $X \cup Y$  that achieves the aforementioned imbalance. For the construction of the ordering on  $X \cup Y$  we consider two cases, either the cardinality of one part is even or none of the cardinalities of the parts are even.

*Case 1.*  $(|X| \bmod 2 = 0) \vee (|Y| \bmod 2 = 0)$ .

W.l.o.g. assume that  $|Y|$  is even. Let  $Y_1$  and  $Y_2$  partition  $Y$  into two sets of equal size. Consider ordering  $\sigma_{X \cup Y} = \sigma_{Y_1} \sigma_X \sigma_{Y_2}$ . In this ordering, the imbalance of any vertex in  $X$  is zero and the imbalance of any vertex in  $Y$  is  $|X|$ . Thus the imbalance of ordering  $\sigma_{X \cup Y}$  is  $I(\sigma_{X \cup Y}) = |X| \cdot |Y|$ .

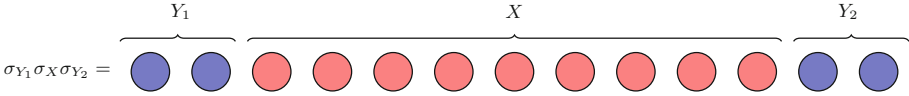


Fig. 1. Example of “sandwiched” ordering for  $G = K_{4,9}$ .

Case 2.  $(|X| \bmod 2 = 1) \wedge (|Y| \bmod 2 = 1)$ .

Let  $y_m \in Y$  be an element of  $Y$ . Let  $X_1$  and  $X_2$  partition  $X$  into two sets such that  $\|X_1| - |X_2|| = 1$  and let  $Y_1$  and  $Y_2$  partition  $Y \setminus \{y_m\}$  into two sets such that  $|Y_1| = |Y_2|$ . Consider the ordering  $\sigma_{X \cup Y} = \sigma_{Y_1} \sigma_{X_1} \sigma_{\{y_m\}} \sigma_{X_2} \sigma_{Y_2}$ . The imbalance of any vertex in  $X \cup \{y_m\}$  is 1 and the imbalance of any vertex in  $Y \setminus \{y_m\}$  is  $|X|$ . Thus the imbalance of the ordering  $\sigma_{X \cup Y}$  is  $I(\sigma_{X \cup Y}) = |X| \cdot |Y| + 1$ .  $\square$

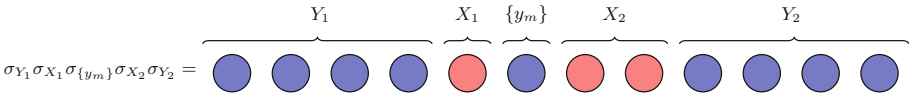


Fig. 2. Example of “pseudo-sandwich” ordering for  $G = K_{3,9}$ .

**Definition 1.** Let  $G = (X, Y, E)$  be a complete bigraph and  $\sigma_{X \cup Y}$  be an arbitrary ordering on  $X \cup Y$ . We define  $L(Y, \sigma_{X \cup Y}) \subseteq [|X| + |Y|]$  to be the positions of the elements of  $Y$  in  $\sigma_{X \cup Y}$ . That is  $L(Y, \sigma_{X \cup Y}) = \{\sigma_{X \cup Y}(y) \mid y \in Y\}$ . Let us denote the elements in  $L(Y, \sigma_{X \cup Y})$  as  $L(Y, \sigma_{X \cup Y}) = \{l_1^{\sigma_{X \cup Y}}, l_2^{\sigma_{X \cup Y}}, \dots, l_{|Y|}^{\sigma_{X \cup Y}}\}$  such that  $l_1^{\sigma_{X \cup Y}} < l_2^{\sigma_{X \cup Y}} < \dots < l_{|Y|}^{\sigma_{X \cup Y}}$ . Additionally, we define  $l_0^{\sigma_{X \cup Y}} = 0$  and  $l_{|Y|+1}^{\sigma_{X \cup Y}} = |X| + |Y| + 1$ . We leave out the superscript in  $l_i^{\sigma_{X \cup Y}}$ , when the ordering is clear from the context.

We define  $L_i^{\sigma_{X \cup Y}}$  to be the vertices of  $X$  between the positions  $l_i$  and  $l_{i+1}$  in ordering  $\sigma_{X \cup Y}$ . Let  $Y = \{y_1, \dots, y_{|Y|}\}$  be an enumeration of the elements in  $Y$  such that  $y_i$  is at position  $l_i$ . We leave out the superscript in  $L_i^{\sigma_{X \cup Y}}$ , when the ordering is clear from the context.

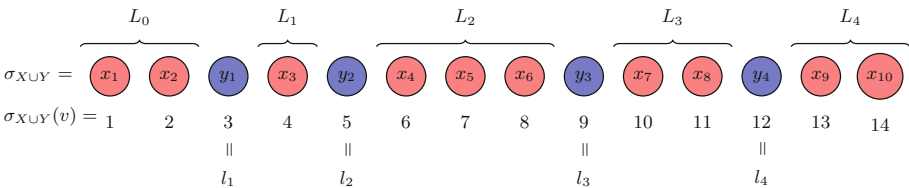


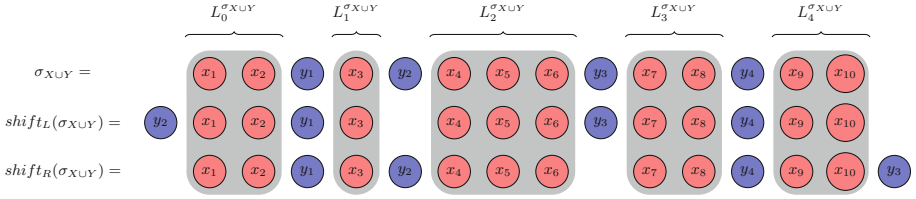
Fig. 3. Visualization of Definition 1 with an ordering  $\sigma_{X \cup Y}$ .

**Definition 2.** Let  $\sigma_{XUY} = \sigma_{L_0} \prod_{i=1}^{\lfloor Y \rfloor} \sigma_{\{y_i\}} \sigma_{L_i}$  be an arbitrary ordering. Let us define

$$\text{shift}_L(\sigma_{XUY}) = \sigma_{\{y_{\lfloor \frac{|Y|}{2} \rfloor}\}} \sigma_{L_0} \left( \prod_{i=1}^{\lfloor \frac{|Y|}{2} \rfloor - 1} \sigma_{\{y_i\}} \sigma_{L_i} \right) \sigma_{L_{\lfloor \frac{|Y|}{2} \rfloor}} \left( \prod_{i=\lfloor \frac{|Y|}{2} \rfloor + 1}^{\lfloor Y \rfloor} \sigma_{\{y_i\}} \sigma_{L_i} \right),$$

$$\text{shift}_R(\sigma_{XUY}) = \sigma_{L_0} \left( \prod_{i=1}^{\lceil \frac{|Y|}{2} \rceil} \sigma_{\{y_i\}} \sigma_{L_i} \right) \sigma_{L_{\lceil \frac{|Y|}{2} \rceil + 1}} \left( \prod_{i=\lceil \frac{|Y|}{2} \rceil + 2}^{\lfloor Y \rfloor} \sigma_{\{y_i\}} \sigma_{L_i} \right) \sigma_{\{y_{\lceil \frac{|Y|}{2} \rceil + 1}\}}.$$

That is,  $\text{shift}_L$  moves vertex  $y_{\lfloor \frac{|Y|}{2} \rfloor}$  to the left most position and  $\text{shift}_R$  moves vertex  $y_{\lceil \frac{|Y|}{2} \rceil + 1}$  to the right most position.



**Fig. 4.** Visualization of the  $\text{shift}_L$  and  $\text{shift}_R$  functions with an ordering  $\sigma_{XUY}$  on the vertices of graph  $G = K_{10,4}$ .

**Lemma 2.** Let  $\sigma_{XUY}$  be an arbitrary imbalance optimal ordering. Let  $\sigma'_{XUY} = \text{shift}_L(\sigma_{XUY})$ . We have that  $I(\sigma_{XUY}) = I(\sigma'_{XUY})$ .

*Proof.* We have that  $I(\sigma_{XUY}, y_{\lfloor \frac{|Y|}{2} \rfloor}) = \left| \left( \sum_{i=0}^{\lfloor \frac{|Y|}{2} \rfloor - 1} |L_i^{\sigma_{XUY}}| \right) - \left( \sum_{i=\lfloor \frac{|Y|}{2} \rfloor}^{\lfloor Y \rfloor} |L_i^{\sigma_{XUY}}| \right) \right|$

and  $I(\sigma'_{XUY}, y_{\lfloor \frac{|Y|}{2} \rfloor}) = \sum_{i=0}^{\lfloor Y \rfloor} |L_i^{\sigma'_{XUY}}| = |X|$ . For each  $0 \leq i \leq \lfloor \frac{|Y|}{2} \rfloor - 1$ , the imbalance of all the vertices in  $L_i^{\sigma'_{XUY}}$  is smaller by 2 in  $\sigma'_{XUY}$ . The imbalance of the remaining vertices remain the same.

Case 3.  $\left( \sum_{i=0}^{\lfloor \frac{|Y|}{2} \rfloor - 1} |L_i^{\sigma_{XUY}}| \right) > \left( \sum_{i=\lfloor \frac{|Y|}{2} \rfloor}^{\lfloor Y \rfloor} |L_i^{\sigma_{XUY}}| \right)$ .

We can express the imbalance of  $I(\sigma'_{XUY})$  as follows:

$$\begin{aligned} I(\sigma'_{XUY}) &= I(\sigma_{XUY}) - 2 \cdot \left( \sum_{i=0}^{\lfloor \frac{|Y|}{2} \rfloor - 1} |L_i^{\sigma_{XUY}}| \right) - I(\sigma_{XUY}, y_{\lfloor \frac{|Y|}{2} \rfloor}) \\ &\quad + I(\sigma'_{XUY}, y_{\lfloor \frac{|Y|}{2} \rfloor}) \end{aligned}$$

$$\begin{aligned}
&= I(\sigma_{X \cup Y}) - 2 \cdot \left( \sum_{i=0}^{\lfloor \frac{|Y|}{2} \rfloor - 1} |L_i^{\sigma_{X \cup Y}}| \right) - \left( \sum_{i=0}^{\lfloor \frac{|Y|}{2} \rfloor - 1} |L_i^{\sigma_{X \cup Y}}| \right) \\
&+ \left( \sum_{i=\lfloor \frac{|Y|}{2} \rfloor}^{|Y|} |L_i^{\sigma_{X \cup Y}}| \right) + \sum_{i=0}^{|Y|} |L_i^{\sigma_{X \cup Y}}| \\
&< I(\sigma_{X \cup Y}).
\end{aligned}$$

Since  $\sigma_{X \cup Y}$  is imbalance optimal, it is not possible that  $I(\sigma'_{X \cup Y}) < I(\sigma_{X \cup Y})$ . Thus this case cannot occur.

$$\text{Case 4. } \left( \sum_{i=0}^{\lfloor \frac{|Y|}{2} \rfloor - 1} |L_i^{\sigma_{X \cup Y}}| \right) \leq \left( \sum_{i=\lfloor \frac{|Y|}{2} \rfloor}^{|Y|} |L_i^{\sigma_{X \cup Y}}| \right).$$

We can express the imbalance of  $I(\sigma'_{X \cup Y})$  as follows:

$$\begin{aligned}
I(\sigma'_{X \cup Y}) &= I(\sigma_{X \cup Y}) - 2 \cdot \left( \sum_{i=0}^{\lfloor \frac{|Y|}{2} \rfloor - 1} |L_i^{\sigma_{X \cup Y}}| \right) - I(\sigma_{X \cup Y}, y_{\lfloor \frac{|Y|}{2} \rfloor}) \\
&+ I(\sigma'_{X \cup Y}, y_{\lfloor \frac{|Y|}{2} \rfloor}) \\
&= I(\sigma_{X \cup Y}) - 2 \cdot \left( \sum_{i=0}^{\lfloor \frac{|Y|}{2} \rfloor - 1} |L_i^{\sigma_{X \cup Y}}| \right) - \left( \sum_{i=\lfloor \frac{|Y|}{2} \rfloor}^{|Y|} |L_i^{\sigma_{X \cup Y}}| \right) \\
&+ \left( \sum_{i=0}^{\lfloor \frac{|Y|}{2} \rfloor - 1} |L_i^{\sigma_{X \cup Y}}| \right) + \sum_{i=0}^{|Y|} |L_i| \\
&= I(\sigma_{X \cup Y}).
\end{aligned}$$

□

**Lemma 3.** Let  $\sigma_{X \cup Y}$  be an arbitrary imbalance optimal ordering. Let  $\sigma'_{X \cup Y} = \text{shift}_R(\sigma_{X \cup Y})$ . We have that  $I(\sigma_{X \cup Y}) = I(\sigma'_{X \cup Y})$ .

*Proof.* Analogous to Lemma 2. □

*Remark 1.* Let both  $|X|$  and  $|Y|$  be odd. Let  $\sigma_{X \cup Y}$  be an arbitrary imbalance optimal ordering. Let  $y_m \in Y$  be the vertex at position  $\lfloor \frac{|Y|}{2} \rfloor$ . We have that  $I(\sigma_{X \cup Y}, y_m) = 1$ . Otherwise  $\sigma_{X \cup Y}$  is not imbalance optimal. This can be shown by a proof by contradiction using Lemma 2 and Lemma 3.

**Theorem 1.** If  $G = (X, Y, E)$  is a complete bigraph, then the minimum imbalance of  $G$  is  $|X| \cdot |Y| + (|X| \bmod 2) \cdot (|Y| \bmod 2)$ .

*Proof.* Let  $\sigma_{X \cup Y}$  be an arbitrary imbalance optimal ordering. Repeatedly apply the functions  $\text{shift}_L$  and  $\text{shift}_R$  on  $\sigma_{X \cup Y}$ , until we have the same ordering as



constructed in Lemma 1. Let us denote the obtained ordering as  $\sigma'_{X \cup Y}$ . Since  $I(\sigma_{X \cup Y}) = I(\sigma'_{X \cup Y})$  by Lemma 2 and Lemma 3, and  $I(\sigma'_{X \cup Y}) = |X| \cdot |Y| + (|X| \bmod 2) \cdot (|Y| \bmod 2)$ , we have that  $I(G) = |X| \cdot |Y| + (|X| \bmod 2) \cdot (|Y| \bmod 2)$ .  $\square$

**Corollary 1.** *By Lemma 2, Lemma 3, and Remark 1, any ordering  $\sigma_{X \cup Y}$  of a complete bigraph  $G = (X, Y, E)$  is imbalance optimal if and only if  $\sigma_{X \cup Y}$  has the following 3 properties:*

1.  $\left( \sum_{i=0}^{\lfloor \frac{|Y|}{2} \rfloor - 1} |L_i^{\sigma_{X \cup Y}}| \right) \leq \left( \sum_{i=\lfloor \frac{|Y|}{2} \rfloor}^{|Y|} |L_i^{\sigma_{X \cup Y}}| \right)$
2.  $\left( \sum_{i=0}^{\lceil \frac{|Y|}{2} \rceil} |L_i^{\sigma_{X \cup Y}}| \right) \geq \left( \sum_{i=\lceil \frac{|Y|}{2} \rceil + 1}^{|Y|} |L_i^{\sigma_{X \cup Y}}| \right)$
3.  $|X|$  and  $|Y|$  are odd  $\Rightarrow I(\sigma_{X \cup Y}, y_m) = 1$ , where  $y_m \in Y$  is at position  $\lfloor \frac{|Y|}{2} \rfloor$ .

The above properties allow us to verify whether any arbitrary ordering is imbalance optimal in  $O(|X| + |Y|)$  time.

**Corollary 2.** *Let  $G = (X, Y, E)$  be a complete bigraph and let  $|X| + |Y| = n$ . Given  $|X|$  and  $|Y|$ , the minimum imbalance  $I(G)$  can be computed in  $\mathcal{O}(\log(n) \cdot \log(\log(n)))$  time by using the formula of Theorem 1. This follows from the fact that the product of two  $k$ -bit integers can be computed in  $\mathcal{O}(k \cdot \log(k))$  time [5].*

## 4 Imbalance on Chained Complete Bipartite Graphs

In this section we shall introduce the chained complete bigraph. We show that the chained complete bigraph is a subclass of PI-bigraphs and how to use the results of Sect. 3 to compute its minimum imbalance efficiently.

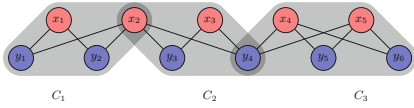
**Definition 3.** *We define  $\mathcal{C}$  to be a family of maximal subsets of the vertices of graph  $G = (V, E)$  that induce a complete bigraph on  $G$ . Additionally, for all edges  $e \in E$  there exists a vertex set  $C_i \in \mathcal{C}$  such that both endpoints of  $e$  are contained in  $C_i$ . That is,  $\mathcal{C} \subseteq \mathcal{P}(V)$  such that:*

$$\begin{aligned} (\forall_{\{u,v\} \in E} \exists_{C_i \in \mathcal{C}} \{u,v\} \subseteq C_i) \wedge (\forall_{C_i \in \mathcal{C}} \exists_{j,k \in \mathbb{N}} G[C_i] = K_{j,k}) \wedge \\ (\forall_{C_i \in \mathcal{C}} \forall_{v \in V \setminus C_i} \forall_{j,k \in \mathbb{N}} G[C_i \cup \{v\}] \neq K_{j,k}), \end{aligned}$$

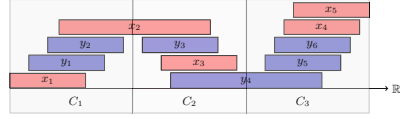
where  $K_{i,j}$  denotes a complete bigraph. We call  $\mathcal{C}$  the maximal complete bigraph components, abbreviated by MCB-components, of  $G$ .

**Definition 4.** *A graph  $G = (X, Y, E)$  is a chained complete bigraph, if  $G$  has a MCB-component family  $\mathcal{C}$  such that we can label  $\mathcal{C} = \{C_1, \dots, C_n\}$  such that consecutive vertex sets  $C_i$  and  $C_{i+1}$  share exactly one vertex and non-consecutive vertex sets  $C_i$  and  $C_j$  share no vertices. Formally,  $(\forall_{1 \leq i < n} |C_i \cap C_{i+1}| = 1) \wedge (\forall_{1 \leq i < j < n} j - i > 1 \implies C_i \cap C_j = \emptyset)$ . We call a vertex that is shared by two consecutive vertex sets of  $\mathcal{C}$  an overlapping vertex.*

For any chained complete bigraph  $G$ , with corresponding MCB-component family  $\mathcal{C}$ , we can create a corresponding PI-bigraph interval representation  $I_G$ . For each  $C_i \in \mathcal{C}$  we create a staircase-shaped set of intervals with the overlapping vertices at the top and bottom.



**Fig. 5.** Example chained complete bigraph. The highlighted areas represent the sets in  $\mathcal{C}$ .



**Fig. 6.** Interval representation of the chained complete bigraph of Fig. 5.

*Remark 2.* By the definition of chained complete bigraph  $G = (X, Y, E)$  we have  $\forall v \in X \cup Y \ 1 \leq |\{C_i \in \mathcal{C} \mid v \in C_i\}| \leq 2$ .

*Remark 3.* By the definition of MCB-components  $\mathcal{C}$  of a chained complete bigraph  $\forall v \in X \cup Y \ N(v) \subseteq \bigcup_{C_j \in \{C_i \in \mathcal{C} \mid v \in C_i\}} C_j$ .

Let  $G = (X, Y, E)$  be a chained complete bigraph with corresponding MCB-component family  $\mathcal{C} = \{C_1, \dots, C_n\}$ . Then we have that:

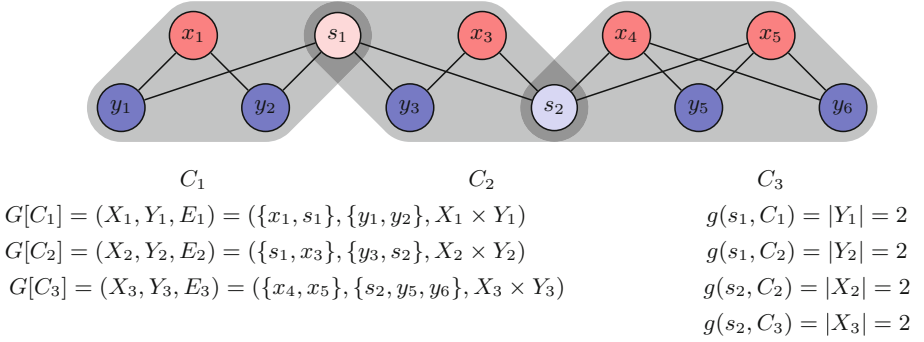
$$I(G) = \sum_{i=1}^n |X_i| \cdot |Y_i| + (|X_i| \bmod 2) \cdot (|Y_i| \bmod 2) - \left( \sum_{i=1}^{n-1} g(s_i, C_i) + g(s_i, C_{i+1}) \right) + \left( \sum_{i=1}^{n-1} |g(s_i, C_i) - g(s_i, C_{i+1})| \right),$$

where  $X_i, Y_i, s_i$ , and the function  $g$  are defined below. First, we introduce additional definitions that are required to understand the proof.

**Definition 5.** Let  $G = (X, Y, E)$  be a chained complete bigraph with corresponding MCB-component family  $\mathcal{C} = \{C_1, \dots, C_n\}$ . We shall use the notation  $G[C_i] = (X_i, Y_i, E_i)$  to denote the graph induced on  $C_i \in \mathcal{C}$ .

**Definition 6.** Let  $G$  be a chained complete bigraph with corresponding MCB-component family  $\mathcal{C} = \{C_1, \dots, C_n\}$ . We label the overlapping vertices of  $\mathcal{C}$  as  $s_i \in C_i \cap C_{i+1}$ , where  $1 \leq i \leq n - 1$ . By the definition of chained complete bigraph, the overlapping vertex  $s_i$  is unique. We define  $S = \{s_i \mid 1 \leq i \leq n - 1\}$ .

**Definition 7.** We define  $g(s_i, C_j)$ , where  $s_i \in S$  and  $C_j \in \mathcal{C}$ , to be the number of neighbors of  $s_i$  in  $C_j$ . Equivalently,  $g(s_i, C_j)$  is the number of vertices in  $C_j$  that do not belong to the same part as  $s_i$ . That is,  $g(s_i, C_j) = |N(s_i) \cap C_j| = \begin{cases} |X_j| & \text{if } s_i \in Y \\ |Y_j| & \text{if } s_i \in X \end{cases}$ .



**Fig. 7.** Illustration of the additional definitions of Sect. 4.

#### 4.1 Proof of the Upper Bound

**Lemma 4.** *Let  $G = (X, Y, E)$  be a chained complete bigraph with corresponding MCB-component family  $\mathcal{C} = \{C_1, \dots, C_n\}$ . Let  $C_i \in \mathcal{C}$  such that  $|X_i| = 1$  or  $|Y_i| = 1$ . W.l.o.g. assume that  $|X_i| = 1$ , then neither overlapping vertices  $s_{i-1}$  nor  $s_i$  can be in  $X_i$ . Formally,  $\forall C_i \in \mathcal{C} (|X_i| = 1 \implies s_{i-1} \notin X_i \wedge s_i \notin X_i) \wedge (|Y_i| = 1 \implies s_{i-1} \notin Y_i \wedge s_i \notin Y_i)$ .*

*Proof.* \* Trivial proof by contradiction. (If not, then  $C_i$  is not maximal.)  $\square$

**Lemma 5.** *Given a chained complete bigraph  $G = (X, Y, E)$  with corresponding MCB-component family  $\mathcal{C} = \{C_1, \dots, C_n\}$ , we have that*

$$I(G) \leq \sum_{i=1}^n |X_i| \cdot |Y_i| + (|X_i| \bmod 2) \cdot (|Y_i| \bmod 2) - \left( \sum_{i=1}^{n-1} g(s_i, C_i) + g(s_i, C_{i+1}) \right) + \left( \sum_{i=1}^{n-1} |g(s_i, C_i) - g(s_i, C_{i+1})| \right).$$

*Proof.* \* The lemma is proven by constructing an ordering  $\sigma_{XUY}$  whose imbalance is equivalent to the above expression. The ordering  $\sigma_{XUY}$  is constructed by creating a subordering for each  $C_i \in \mathcal{C}$  separately and concatenating those suborderings. The suborderings are created in a similar fashion as the orderings in the proof of Lemma 1.  $\square$

#### 4.2 Proof of the Lower Bound

**Remark 4.** The imbalance of  $v \in (X \cup Y) \setminus S$  is only influenced by the vertices in  $C_i$ . That is,  $\forall C_i \in \mathcal{C} \forall v \in C_i \setminus S I(v, \sigma_{XUY}, G) = I(v, \sigma_{XUY}^{C_i}, G[C_i])$ .

**Remark 5.** The imbalance of  $s_i$  is only influenced by the vertices in  $C_i \cup C_{i+1}$ . That is,  $\forall s_i \in S I(s_i, \sigma_{XUY}, G) = I(s_i, \sigma_{XUY}^{C_i \cup C_{i+1}}, G[C_i \cup C_{i+1}])$ .

**Lemma 6.** *Let  $G = (X, Y, E)$  be a chained complete bigraph with corresponding MCB-component family  $\mathcal{C} = \{C_1, \dots, C_n\}$ . For any arbitrary ordering  $\sigma_{XUY}$  it holds that*

$$\begin{aligned} & I(s_{n-1}, \sigma_{XUY}^{C_{n-1} \cup C_n}, G[C_{n-1} \cup C_n]) - I(s_{n-1}, \sigma_{XUY}^{C_{n-1}}, G[C_{n-1}]) \\ & - I(s_{n-1}, \sigma_{XUY}^{C_n}, G[C_n]) \\ & \geq |g(s_{n-1}, C_{n-1}) - g(s_{n-1}, C_n)| - g(s_{n-1}, C_{n-1}) - g(s_{n-1}, C_n). \end{aligned}$$

*Proof.* The expression  $|g(s_{n-1}, C_{n-1}) - g(s_{n-1}, C_n)| - g(s_{n-1}, C_{n-1}) - g(s_{n-1}, C_n)$  takes two possible values depending on the sign of  $g(s_{n-1}, C_{n-1}) - g(s_{n-1}, C_n)$ . Either the above expression is equivalent to  $-2 \cdot g(s_{n-1}, C_n)$  or  $-2 \cdot g(s_{n-1}, C_{n-1})$ . To relate the expressions in the inequality, we shall denote the number of neighbors of  $s_{n-1}$  to its left and to its right in  $C_n$  and  $C_{n-1}$  in ordering  $\sigma_{XUY}$  as:

$$\begin{aligned} l_1 &= |\{v \in C_{n-1} \cap N(s_{n-1}) \mid v <_{\sigma_{XUY}} s_{n-1}\}|; \\ l_2 &= |\{v \in C_n \cap N(s_{n-1}) \mid v <_{\sigma_{XUY}} s_{n-1}\}|; \\ r_1 &= |\{v \in C_{n-1} \cap N(s_{n-1}) \mid v >_{\sigma_{XUY}} s_{n-1}\}|; \\ r_2 &= |\{v \in C_n \cap N(s_{n-1}) \mid v >_{\sigma_{XUY}} s_{n-1}\}|. \end{aligned}$$

Using the above definitions, we rewrite the expression on the left-hand side of the inequality as follows:

$$\begin{aligned} & I(s_{n-1}, \sigma_{XUY}^{C_{n-1} \cup C_n}, G[C_{n-1} \cup C_n]) - I(s_{n-1}, \sigma_{XUY}^{C_{n-1}}, G[C_{n-1}]) \\ & - I(s_{n-1}, \sigma_{XUY}^{C_n}, G[C_n]) \\ & = |l_1 - r_1 + l_2 - r_2| - |l_1 - r_1| - |l_2 - r_2|. \end{aligned}$$

According to the signs of  $l_1 - r_1$  and  $l_2 - r_2$ , consider the following cases:

$$\begin{aligned} & (++) \quad l_1 - r_1 \geq 0 \text{ and } l_2 - r_2 \geq 0; \quad (+-) \quad l_1 - r_1 \geq 0 \text{ and } l_2 - r_2 < 0; \\ & (-+) \quad l_1 - r_1 < 0 \text{ and } l_2 - r_2 \geq 0; \quad (--) \quad l_1 - r_1 < 0 \text{ and } l_2 - r_2 < 0 \end{aligned}$$

For the cases  $(++)$  and  $(--)$ , we have that  $|l_1 - r_1 + l_2 - r_2| - |l_1 - r_1| - |l_2 - r_2| = 0$ . Thus, in the cases  $(++)$  and  $(--)$ , it holds that

$$\begin{aligned} & |l_1 - r_1 + l_2 - r_2| - |l_1 - r_1| - |l_2 - r_2| = 0 \\ & \geq |g(s_{n-1}, C_{n-1}) - g(s_{n-1}, C_n)| - g(s_{n-1}, C_{n-1}) - g(s_{n-1}, C_n). \end{aligned}$$

This follows from the fact that  $-2 \cdot g(s_{n-1}, C_n) \leq 0$  and  $-2 \cdot g(s_{n-1}, C_{n-1}) \leq 0$ . For the cases  $(+-)$  and  $(-+)$ , we have that

$$|l_1 - r_1 + l_2 - r_2| - |l_1 - r_1| - |l_2 - r_2| = \begin{cases} 2(l_2 - r_2) & \text{if } (+-) \wedge l_1 - r_1 + l_2 - r_2 \geq 0 \\ 2(r_2 - l_2) & \text{if } (-+) \wedge l_1 - r_1 + l_2 - r_2 < 0 \\ 2(r_1 - l_1) & \text{if } (+-) \wedge l_1 - r_1 + l_2 - r_2 < 0 \\ 2(l_1 - r_1) & \text{if } (-+) \wedge l_1 - r_1 + l_2 - r_2 \geq 0 \end{cases}.$$

Observe that, by the definitions of  $l_1$ ,  $r_1$ ,  $l_2$ ,  $r_2$ , and function  $g$ , we have  $l_1 + r_1 = g(s_{n-1}, C_{n-1})$  and  $l_2 + r_2 = g(s_{n-1}, C_n)$ . Using the above remark and case distinction, we derive that in the cases  $(+-)$  and  $(-+)$  it holds that

$$\begin{aligned} & |l_1 - r_1 + l_2 - r_2| - |l_1 - r_1| - |l_2 - r_2| \\ \geq & |g(s_{n-1}, C_{n-1}) - g(s_{n-1}, C_n)| - g(s_{n-1}, C_{n-1}) - g(s_{n-1}, C_n). \end{aligned}$$

□

**Lemma 7.** *Given a chained complete bigraph  $G = (X, Y, E)$  with corresponding MCB-component family  $\mathcal{C} = \{C_1, \dots, C_n\}$ , we have that*

$$\begin{aligned} I(G) \geq & \sum_{i=1}^n |X_i| \cdot |Y_i| + (|X_i| \bmod 2) \cdot (|Y_i| \bmod 2) \\ & - \left( \sum_{i=1}^{n-1} g(s_i, C_i) + g(s_i, C_{i+1}) \right) + \left( \sum_{i=1}^{n-1} |g(s_i, C_i) - g(s_i, C_{i+1})| \right). \end{aligned}$$

*Proof.* <sup>\*1</sup>We shall prove that the imbalance of any arbitrary ordering  $\sigma_{XUY}$  on the vertex set  $X \cup Y$  is bounded from below by the above expression by induction on  $|\mathcal{C}| = n$ .

- Base Case ( $n = 0 \vee n = 1$ ):

By the definition of MCB-components  $\mathcal{C}$ , the graph  $G$  is an empty graph or a complete bigraph. Thus, by Theorem 1, the lemma holds for the base case.

- Induction step ( $n > 1$ ):

Let  $G = (X, Y, E)$  be a chained complete bigraph with corresponding MCB-component family  $\mathcal{C} = \{C_1, \dots, C_{k+1}\}$ . Let us define  $\mathcal{C} \setminus C_{k+1} =$

$\bigcup_{C_i \in \mathcal{C} \setminus \{C_{k+1}\}} C_i$ . We write the imbalance of  $\sigma_{XUY}$  as follows:

$$\begin{aligned} I(\sigma_{XUY}) = & I(\sigma_{XUY}^{\mathcal{C} \setminus C_{k+1}}, G[\mathcal{C} \setminus C_{k+1}]) + I(\sigma_{XUY}^{C_{k+1}}, G[C_{k+1}]) \\ & - I(s_k, \sigma_{XUY}^{C_k}, G[C_k]) - I(s_k, \sigma_{XUY}^{C_{k+1}}, G[C_{k+1}]) \\ & + I(s_k, \sigma_{XUY}^{C_k \cup C_{k+1}}, G[C_k \cup C_{k+1}]) \end{aligned} \quad (1)$$

$$\begin{aligned} \geq & \left( \sum_{i=1}^k |X_i| \cdot |Y_i| + (|X_i| \bmod 2) \cdot (|Y_i| \bmod 2) \right) \\ & - \left( \sum_{i=1}^{k-1} g(s_i, C_i) + g(s_i, C_{i+1}) \right) + \left( \sum_{i=1}^{k-1} |g(s_i, C_i) - g(s_i, C_{i+1})| \right) \\ & + |X_{k+1}| \cdot |Y_{k+1}| + (|X_{k+1}| \bmod 2) \cdot (|Y_{k+1}| \bmod 2) \\ & - I(s_k, \sigma_{XUY}^{C_k}, G[C_k]) - I(s_k, \sigma_{XUY}^{C_{k+1}}, G[C_{k+1}]) \\ & + I(s_k, \sigma_{XUY}^{C_k \cup C_{k+1}}, G[C_k \cup C_{k+1}]) \end{aligned} \quad (2)$$

$$\begin{aligned} \geq & \left( \sum_{i=1}^{k+1} |X_i| \cdot |Y_i| + (|X_i| \bmod 2) \cdot (|Y_i| \bmod 2) \right) \\ & - \left( \sum_{i=1}^{k-1} g(s_i, C_i) + g(s_i, C_{i+1}) \right) + \left( \sum_{i=1}^{k-1} |g(s_i, C_i) - g(s_i, C_{i+1})| \right) \end{aligned}$$

<sup>1</sup> The proofs marked with an \* are provided in detail in the full version of the paper.

$$\begin{aligned}
 & -g(s_k, C_{k+1}) - g(s_k, C_k) + |g(s_k, C_k) - g(s_k, C_{k+1})| \tag{3} \\
 & = \left( \sum_{i=1}^{k+1} |X_i| \cdot |Y_i| + (|X_i| \bmod 2) \cdot (|Y_i| \bmod 2) \right) \\
 & - \left( \sum_{i=1}^k g(s_i, C_i) + g(s_i, C_{i+1}) \right) + \left( \sum_{i=1}^k |g(s_i, C_i) - g(s_i, C_{i+1})| \right),
 \end{aligned}$$

where Eq. (1) follows from Remark 4 and Remark 5, Eq. (2) follows from the induction hypothesis, and Eq. (3) follows from Lemma 6.  $\square$

**Theorem 2.** *Let  $G = (X, Y, E)$  be a chained complete bigraph with corresponding MCB-component family  $\mathcal{C} = \{C_1, \dots, C_n\}$ . We have that*

$$\begin{aligned}
 I(G) &= \sum_{i=1}^n |X_i| \cdot |Y_i| + (|X_i| \bmod 2) \cdot (|Y_i| \bmod 2) \\
 & - \left( \sum_{i=1}^{n-1} g(s_i, C_i) + g(s_i, C_{i+1}) \right) + \left( \sum_{i=1}^{n-1} |g(s_i, C_i) - g(s_i, C_{i+1})| \right).
 \end{aligned}$$

*Proof.* Follows from Lemma 5 and Lemma 7.  $\square$


**Corollary 3.** *Let  $G = (X, Y, E)$  be a chained complete bigraph with corresponding MCB-component family  $\mathcal{C} = \{C_1, \dots, C_n\}$  and let  $|X| + |Y| = m$ . Given  $\{|X_1|, \dots, |X_n|\}$ ,  $\{|Y_1|, \dots, |Y_n|\}$ , and  $\{s_1 \in X, \dots, s_{n-1} \in X\}$ , the imbalance of  $G$  can be computed in  $\mathcal{O}(n \cdot \log(m) \cdot \log(\log(m)))$  time. By applying a similar reasoning as in Theorem 2, we verify the correctness of this corollary.*

## References

1. Biedl, T., Chan, T., Ganjali, Y., Hajiaghayi, M.T., Wood, D.R.: Balanced vertex-orderings of graphs. *Discrete Appl. Math.* **148**(1), 27–48 (2005). <https://doi.org/10.1016/j.dam.2004.12.001>, <http://www.sciencedirect.com/science/article/pii/S0166218X04003828>
2. Díaz, J., Petit, J., Serna, M.: A survey of graph layout problems. *ACM Comput. Surv. (CSUR)* **34**(3), 313–356 (2002)
3. Gorzny, J.: Computing imbalance-minimal orderings for bipartite permutation graphs and threshold graphs. In: Wu, W., Zhang, Z. (eds.) *COCOA 2020*. LNCS, vol. 12577, pp. 766–779. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-64843-5\\_52](https://doi.org/10.1007/978-3-030-64843-5_52)
4. Gorzny, J., Buss, J.F.: Imbalance, cutwidth, and the structure of optimal orderings. In: Du, D.-Z., Duan, Z., Tian, C. (eds.) *COCOON 2019*. LNCS, vol. 11653, pp. 219–231. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-26176-4\\_18](https://doi.org/10.1007/978-3-030-26176-4_18)
5. Harvey, D., van der Hoeven, J.: Integer multiplication in time  $\mathcal{o}(n \log n)$ . *Ann. Math.* **193**(2), 563–617 (2021). <https://www.jstor.org/stable/10.4007/annals.2021.193.2.4>
6. Kára, J., Kratochvíl, J., Wood, D.R.: On the complexity of the balanced vertex ordering problem. In: Wang, L. (ed.) *COCOON 2005*. LNCS, vol. 3595, pp. 849–858. Springer, Heidelberg (2005). [https://doi.org/10.1007/11533719\\_86](https://doi.org/10.1007/11533719_86)
7. Wood, D.: Minimising the number of bends and volume in three-dimensional orthogonal graph drawings with a diagonal vertex layout. *Algorithmica (New York)* **39** (2002). <https://doi.org/10.1007/s00453-004-1091-4>



# New Algorithms for a Simple Measure of Network Partitioning

Xueyang Zhao, Binghao Yan, and Peng Zhang<sup>(✉)</sup> 

School of Software, Shandong University, Jinan, Shandong 250101, China  
{algyoung, algyanbh}@mail.sdu.edu.cn, algzhang@sdu.edu.cn

**Abstract.** Partitioning a network into  $k$  pieces is a fundamental problem in network science. A simple measure of partitioning a network is provided by the Max  $k$ -Uncut problem. Given an  $n$ -vertex undirected graph  $G$  with nonnegative weights defined on edges, and a positive integer  $k$ , the Max  $k$ -Uncut problem asks to find a  $k$ -partition of the vertices of  $G$  to maximize the total weight of edges that are not in the cut. This problem is the complement of the classic Min  $k$ -Cut problem, and has close relation to many combinatorial optimization problems, including the famous Densest  $k$ -Subgraph problem. In this paper, we propose a greedy approximation algorithm for the Max  $k$ -Uncut problem with performance ratio  $1 - \frac{2(k-1)}{n}$ . The algorithm is very simple, which consists of only  $k - 1$  min cut computations. The algorithm has fast running time  $O(kn^2)$  and is hence implementable. The experimental results show that the algorithm has excellent practical performance.

**Keywords:** Max  $k$ -Uncut · Network partitioning · Graph algorithm · Approximation algorithm · Combinatorial optimization

## 1 Introduction

In network science, partitioning a network into communities is a very fundamental problem, which receives much attention from the researchers coming from different areas such as network science, computer science, operations research, and artificial intelligence, etc. Intuitively, a *community* in a network is a set of nodes which have close internal connections. The communities of a network can be overlapping or non-overlapping. In this paper we focus on the *non-overlapping* community structure of a network. This is equivalent to, in terms of graph theory, the *partitioning* of vertices of a graph. The non-overlapping community structure is sometimes also called a *flat clustering* (of nodes in a network). We study the Max  $k$ -Uncut problem, which aims to partition a graph into  $k$  pieces. The problem is given in Definition 1.

**Definition 1. The Max  $k$ -Uncut Problem, in terms of partition.**

Instance: An  $n$ -vertex undirected graph  $G = (V, E)$ , and an integer  $k > 0$ .

Goal: Find a partition  $\mathcal{P} = \{V_1, V_2, \dots, V_k\}$  of  $V$  such that the number of edges not cut is maximized.

In Definition 1, the partition  $\mathcal{P} = \{V_1, V_2, \dots, V_k\}$  is called a  $k$ -partition, which partitions the vertices of  $G$  into  $k$  *non-empty* pieces. Given a partition  $\mathcal{P}$  and an edge  $e = (u, v)$ , if the two endpoints  $u$  and  $v$  of  $e$  are in different pieces of  $\mathcal{P}$ , then we say that  $e$  is cut by  $\mathcal{P}$ . Otherwise (that is,  $u$  and  $v$  are in the same piece),  $e$  is not cut by  $\mathcal{P}$ .

Whereas there are many heuristics and algorithms that are proposed to find the community structure of a network, the Max  $k$ -Uncut problem still provides a very simple measure for the community structure, that is, the number of uncut edges resulted from the structure. More introduction about the methods of finding community structure can be found in Sect. 1.1.

Now we introduce the second origin of the Max  $k$ -Uncut problem. It is well-known that homophily is a fundamental law governing the evolution of networks. The *homophily law* says that in a network nodes are likely to connect to nodes with the same or similar attributes. This is just like a proverb which says that “birds of a feather flock together”. Viewing attributes as colors, we can define the happiness of edges as follows [33,36]. Given an edge  $e = (u, v)$ , if  $u$  and  $v$  has the same color, then we say that edge  $e$  is *happy*. Otherwise, edge  $e$  is *unhappy*. In terms of homophily, the Max  $k$ -Uncut problem can be defined as in Definition 2 [35].

**Definition 2. The Max  $k$ -Uncut Problem, in terms of homophily.**

Instance: An  $n$ -vertex undirected graph  $G = (V, E)$ , and a color set  $C = \{1, 2, \dots, k\}$ .

Goal: Color the vertices in  $G$  in  $k$  colors such that the number of happy edges is maximized.

In practice there are many networks whose nodes may have attributes with unknown values. For example, for a paper the area (or field) to which it belongs is a natural attribute. However, in a paper citation network it is very common that a paper is not marked with its area. The Max  $k$ -Uncut problem is useful to recover the attribute values of nodes in a network when the attributes values are unknown.

Given a colored graph  $G$ , if its vertices are colored in *exactly*  $k$  colors, then we say that the coloring of  $V(G)$  is a  $k$ -coloring. It is easy to see that a  $k$ -partition of  $V(G)$  and a  $k$ -coloring of  $V(G)$  are in fact equivalent. This means that Definition 1 and Definition 2 are two equivalent definitions of Max  $k$ -Uncut.

The weighted Max  $k$ -Uncut problem can be easily defined as follows. Each edge  $e$  has a nonnegative weight  $w(e)$ , and the goal of the problem is to maximize the total weight of happy edges.

The Max  $k$ -Uncut problem arose in the study of network science recently [35]. There are many methods so far to find communities in a network. Some of them are heuristics, for example, the method based on edge betweenness [9, 11, 23], and the label propagation method [27], etc. Some of them are to optimize an objective function, for example, the modularity-based method [6, 24, 25], the hierarchical clustering method [8], the structural entropy method [20, 21], and the  $k$ -means clustering method [2, 16, 22], etc.



Although so many methods for finding communities have been proposed, the problem of finding communities itself has not been well solved yet. This is partially because there is no a rigorous mathematical definition for community so far. Different application scenarios have their own communities. Perhaps it is hard to give a general definition for community to fit all the application scenarios.

The Max  $k$ -Uncut problem still provide a simple measure for finding communities in a network. The new measure of Max  $k$ -Uncut is just the number of happy edges. It is a simple measure, especially in contrast with the complicated measures such as, e.g., modularity [25] and structural entropy [20]. Methods for finding communities that are similar to Max  $k$ -Uncut include the  $k$ -Means problem [22], the  $k$ -Center problem [14], and the  $k$ -Median problem [15], etc.

Given a graph  $G = (V, E)$ , we always use  $n$  to denote its vertex number, and  $m$  its edge number. Given an optimization problem, we use  $OPT$  to denote the optimum value of (the instance of) the problem.

## 1.1 Related Work

The Max  $k$ -Uncut problem was proposed by Zhang et al. [35], where a randomized  $(1 - \frac{k}{n})^2$ -approximation algorithm and a greedy  $(1 - \frac{2(k-1)}{n})$ -approximation algorithm were given for the problem. [35] proved that if the Densest  $k$ -Subgraph problem can be approximated within  $\alpha$ , the Max  $k$ -Uncut problem can be approximated within  $\Omega(\frac{1}{2}\alpha)$ . The current best approximation ratio for Densest  $k$ -Subgraph is  $O(n^{1/4-\epsilon})$  [3]. Moreover, [35] proved that Max  $k$ -Uncut and the Densest  $k$ -subgraph problem are actually equivalently in terms of approximability (up to a constant factor 2). Very recently, Zhang et al. [34] achieved an expected approximation ratio  $\frac{1}{2}(1 + (\frac{n-k}{n})^2)$  for Max  $k$ -Uncut using the LP-rounding plus greed strategy.

Choudhurya et al. [7] proposed the capacited Max  $k$ -Uncut problem. Wu et al. [31] studied the balanced Max 3-Uncut problem, in which an input graph is partitioned into 3 equal-sized parts so that the total weight of happy edges is maximized.

The algorithmic study of homophyly of networks started only recently [33, 36] and has attracted much attention from researchers [4, 5, 18, 19, 26]. The algorithmic aspects of homophyly include two interesting coloring problems, i.e., the Maximum Happy Edges (MHE) problem and the Maximum Happy Vertices (MHV) problem. Given a partially colored graph (namely, only part of vertices are colored), the MHE problem asks to color all the uncolored vertices so that the number of happy edges is maximized. Similarly, the MHV problem wants to maximize the number of happy vertices, where a vertex is happy if the vertex and its neighbours are all colored in the same color. It is easy to see that the Max  $k$ -Uncut problem is closely related to MHE.

It is interesting that Max  $k$ -Uncut has rich connection to many problems in combinatorial optimization. Max  $k$ -Uncut is just the complement of the classic Min  $k$ -Cut problem [13]. The Min  $k$ -Cut problem asks for a  $k$ -partition such that the total weight of cut edges is minimized. The current best approximation ratio

of Min  $k$ -Cut is 2 [29]. Another closely related problem is Max  $k$ -Cut [10], which asks to find a  $k$ -partition such that the total weight of cut edges is maximized. When  $k = 2$ , Max  $k$ -Cut is just the famous Max Cut problem [12].

In literature, the “uncut” problems have also been studied extensively. Besides Max  $k$ -Uncut, three examples are Min Uncut [1], Multiway Uncut [17, 36], and the complement of Min Bisection [32].

## 1.2 Our Results

Our main result is a new approximation algorithm for the Max  $k$ -Uncut problem, which is called Algorithm MG (see Algorithm 2.1) in the paper. Algorithm MG is very simple. It consists of only  $k - 1$  min cut computations. The algorithm repeatedly separates one of the current subgraphs into two subgraphs by computing the minimum cut of that graph, until  $k$  subgraphs are obtained. Algorithm MG is a greedy algorithm since it always cuts the subgraph with the lightest minimum cut value among all the current subgraphs. The time complexity of Algorithm MG is  $O(kn^2)$ , implying that it is a practical algorithm.

We prove that the approximation ratio of Algorithm MG is  $1 - \frac{2(k-1)}{n}$ . It is interesting that Algorithm MG has the same approximation ratio as the greedy algorithm given by [35] (called Algorithm G in this paper). The quality of ratio  $1 - \frac{2(k-1)}{n}$  depends on the part number  $k$  and the vertex number  $n$ . When  $k$  is small, the ratio is good. When  $k$  gets larger, the ratio becomes worse. Note that when  $k \leq \frac{n}{4} + 1$ , the ratio is always  $\geq \frac{1}{2}$ . In many applications such as finding the community structure of a large scale network, the part number  $k$  is usually much smaller than the vertex number  $n$ . So, it is expected that Algorithm MG will behave well in these applications.

Algorithm G produces a  $k$ -partition by picking the first  $k - 1$  vertices with the smallest degrees. The  $k$ -partition of Algorithm G then consists of  $k - 1$  singleton sets and one set of the remaining  $n - (k - 1)$  vertices. Note that this is a special structure, that we call *big-endian*. Obviously, a big-endian partition represents an extreme case among all possible partitions. It is difficult to imagine that a partition of a complex network from real world (e.g., social network, paper citation network) is big-endian. In contrast, the  $k$ -partition produced by Algorithm MG is more natural. It is obtained by  $k - 1$  min cuts, and hence not big-endian in general.

Besides Algorithm MG, we also devise two more algorithms for the Max  $k$ -Uncut problem, that is, Algorithm MAS and Algorithm BP. Algorithm MAS uses the maximum adjacency search strategy to find a  $k$ -partition of a graph. The maximum adjacency search is a common strategy to solve a class of optimization problems. For example, it was used to solve the Minimum Cut problem [30] and the Densest  $k$ -Subgraph problem [28]. Algorithm BP is like Algorithm MG, except that in each iteration Algorithm BP always finds the subgraph which comes first to cut, instead of finding the subgraph with the lightest minimum cut value (as in Algorithm MG).

We test Algorithm MG, as well as other four algorithms (that is, Algorithms MAS, BP, R and G) on randomly generated graphs. We use three graph gener-

ating models to generate the test instances. These models are the  $G[n, p]$  model, the configuration model, and the preferential attachment model. The experimental results show that Algorithm MG demonstrates extremely well practical performance, and is strictly better than the other four algorithms tested in the paper.

## 2 The Greedy Cut Algorithm

### 2.1 The Algorithm

When  $k = 2$ , the Max 2-Uncut problem can be solved optimally by finding the minimum cut of the graph. This suggests a very natural greedy algorithm for Max 2-Uncut when  $k > 2$ , given as Algorithm 2.1. Initially, the graph is partitioned into two parts by computing a minimum cut. We want the total weight of happy edges is as large as possible. So, in the next step, we just choose the smaller cut from the two minimum cuts for the two parts to get a 3-partition for the original graph  $G$ . The above process is repeated for  $k - 1$  times to get a  $k$ -partition for graph  $G$ .

---

#### Algorithm 2.1. Greedy Cut (Algorithm MG)

---

**Input:** A graph  $G = (V, E)$ , and an integer  $k > 1$ .

**Output:** A  $k$ -partition of  $G$ .

- 1  $\mathcal{P} \leftarrow \{V\}$ .
  - 2 **for**  $i \leftarrow 1$  **to**  $k - 1$  **do**
  - 3   Compute a minimum cut of the induced graph  $G[S]$  for each part  $S \in \mathcal{P}$  with  $|S| \geq 2$ .
  - 4   Let  $\hat{S} \in \mathcal{P}$  be the part with the smallest minimum cut value in Step 3, and  $(S', S'')$  be the minimum cut on graph  $G[\hat{S}]$ .
  - 5   Remove  $\hat{S}$  from  $\mathcal{P}$ , then add  $S'$  and  $S''$  to  $\mathcal{P}$ .
  - 6 **end for**
  - 7 **return**  $\mathcal{P}$ .
- 

For the sake of simplicity, we call Algorithm 2.1 as Algorithm MG, where MG means multiple greed.

As stated before Algorithm MG, we need only to deal with Max  $k$ -Uncut for  $k \geq 3$ . So, we have the following Assumption 1.

**Assumption 1.** *For the Max  $k$ -Uncut problem, we may assume that  $k \geq 3$ . Furthermore, we may assume that  $n \geq k + 1$  since the case  $n = k$  is trivial for Max  $k$ -Uncut.*

### 2.2 Analysis

In this section we will prove that Algorithm MG is a  $(1 - \frac{2(k-1)}{n})$ -approximation algorithm for the Max  $k$ -Uncut problem.

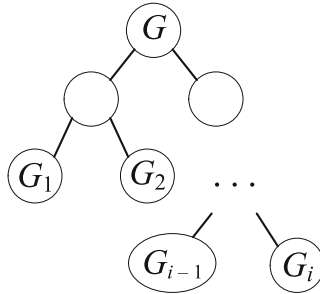
Let  $v$  be a vertex. We use  $wd(v)$  to denote the *weighted degree* of vertex  $v$ , that is,

$$wd(v) = w(\delta(v)),$$

where  $\delta(v)$  is the set of all edges that attach to vertex  $v$ , and  $w(\delta(v))$  is the total weight of these edges. One can easily see that if every edge has unit weight in the graph, then  $wd(v)$  is just the common degree  $d(v)$ . Without loss of generality, we assume that the vertices of graph  $G$  are numbered as  $v_1, v_2, \dots, v_n$  so that

$$wd(v_1) \leq wd(v_2) \leq \dots \leq wd(v_n). \tag{1}$$

The `for` loop of Algorithm MG consists of  $k - 1$  iterations. At the end of the first iteration, the algorithm generates a 2-partition of graph  $G$ . When the second iteration is finished, the algorithm generates a 3-partition. In general, at the end of the  $(i - 1)$ -th iteration, the algorithm generates an  $i$ -partition of graph  $G$ . Moreover, the generated parts in the execution process of Algorithm MG can be organized in a tree, which is called a *partition tree*. Figure 1 shows an imaginary partition tree at the end of the  $(i - 1)$ -th iteration of Algorithm MG. Each node in the partition tree denotes a part  $S$  of the current partition. Let the node corresponding to part  $S$  also denote the induced subgraph  $G[S]$ . So, each node of the partition tree equivalently denotes a subgraph induced on that part. The root of the tree denotes the original graph  $G$ . The  $i$ -partition found at the end of the  $(i - 1)$ -th iteration just consists of the leaves of the partition tree. Note that the partition tree is a regular binary tree, that is, each internal node of the tree has just two children.



**Fig. 1.** An imaginary partition tree at the end of the  $(i - 1)$ -th iteration.

Let us consider the  $i$ -th iteration ( $1 \leq i \leq k - 1$ ) of Algorithm MG. The algorithm will compute a minimum cut for each part in the current partition  $\mathcal{P}$ . Consequently,  $i$  cuts are found in this iteration, as partition  $\mathcal{P}$  contains  $i$  parts. Let  $C_i = \delta(S', S'')$  (see Step 4 of Algorithm MG) be the cut of the minimum value among these  $i$  cuts, where  $\delta(S', S'')$  denotes the set of all edges lying between  $S'$  and  $S''$  (that is, all edges with one endpoint in  $S'$  and the other endpoint in  $S''$ ). Our key observation is Lemma 1.

**Lemma 1.** *Let  $C_i$  and  $v_i$  be defined as above. For each  $1 \leq i \leq k - 1$ , we have*

$$w(C_i) \leq wd(v_i).$$

Before giving the proof of Lemma 1, let us see two specific cases to have a warm-up preparation of the proof.

Lemma 1 obviously holds for the first iteration. When  $i = 1$ ,  $C_1$  is the minimum cut on graph  $G$ . As  $(v_1, V \setminus \{v_1\})$  is obviously a cut of  $G$ , its value, namely, the weighted degree  $wd(v_1) = w(\delta(v_1, V \setminus \{v_1\}))$ , must be an upper bound of  $w(C_1)$ .

Then consider  $i = 2$ . At the beginning of the second iteration, partition  $\mathcal{P}$  contains two parts,  $S_1$  and  $S_2$  to say. Let  $G_1 = G[S_1]$  and  $G_2 = G[S_2]$  be the corresponding induced subgraphs. For a graph  $G$ , let  $gc^*(G)$  denote the value of the minimum (global) cut of  $G$ . If  $G$  contains only one vertex, then there is no any cut for  $G$  and we define  $gc^*(G) = \infty$  in this case.

First we consider the vertex  $v_2$ . Suppose that  $v_2$  is in  $G_{j_2}$  for some  $j_2 \in \{1, 2\}$ . We will consider two cases, that is, the case  $|V(G_{j_2})| \geq 2$  and the case  $|V(G_{j_2})| = 1$ .

If  $|V(G_{j_2})| \geq 2$ , we have

$$w(C_2) \leq gc^*(G_{j_2}) \leq wd_{G_{j_2}}(v_2) \leq wd(v_2),$$

where  $wd_{G_{j_2}}(v_2)$  denotes the weighted degree of  $v_2$  on graph  $G_{j_2}$ . The first inequality stands since  $C_2$  is the cut with the minimum value among the two minimum cuts for  $G_1$  and  $G_2$ . The second inequality stands since  $(v_2, V(G_{j_2}) \setminus \{v_2\})$  is a global cut of  $G_{j_2}$  while  $gc^*(G_{j_2})$  is the value of the minimum (global) cut of  $G_{j_2}$ , which is no more than the weighted degree of any vertex in  $G_2$ . The third inequality stands since  $G_{j_2}$  is a subgraph of  $G$ , and  $wd(v_2)$  is just  $wd_G(v_2)$ .

If  $|V(G_{j_2})| = 1$  (i.e.,  $G_{j_2}$  contains only one vertex, which is just  $v_2$ ), then the minimum cut of  $G_2$  is not well-defined. Recall from Step 3 of Algorithm MG that Algorithm G would not “cut” such a graph. In this case we have to turn to consider the vertex  $v_1$ . Since  $v_2$  is the only vertex in  $G_{j_2}$ ,  $v_1$  must be in  $G_{j_1}$  where  $\{j_1\} = \{1, 2\} \setminus \{j_2\}$ . We confirm that  $|V(G_{j_1})|$  must be  $\geq 2$ , since otherwise graph  $G$  contains only two vertices, which is absurd. So, we have

$$w(C_2) \leq gc^*(G_{j_1}) \leq wd_{G_{j_1}}(v_1) \leq wd(v_1) \leq wd(v_2).$$

So far, we have proved that Lemma 1 holds for the second iteration.

*Proof (Proof of Lemma 1).* Let  $G_1, G_2, \dots, G_i$  be the subgraphs formed by the partition  $\mathcal{P}$  at the start of the  $i$ -th iteration of Algorithm MG.

First we consider vertex  $v_i$ . Suppose  $v_i$  is in  $G_{j_i}$  for some  $1 \leq j_i \leq i$ . We distinguish two cases:  $|V(G_{j_i})| \geq 2$  and  $|V(G_{j_i})| = 1$ .

**Case 1:**  $|V(G_{j_i})| \geq 2$ . In this case,  $gc^*(G_{j_i})$  is well-defined and we have

$$w(C_i) \leq gc^*(G_{j_i}) \leq wd_{G_{j_i}}(v_i) \leq wd(v_i),$$

where the first inequality holds since  $C_i$  is the cut with the minimum value among all the  $i$  minimum cuts, the second inequality holds since  $gc^*(G_{j_i})$  is the value of the minimum (global) cut and  $(v_i, V(G_{j_i}) \setminus \{v_i\})$  is a global cut, and the third inequality holds since  $G_{j_i}$  is a subgraph of  $G$ .

**Case 2:**  $|V(G_{j_i})| = 1$ . In this case,  $gc^*(G_{j_i})$  is not defined. We have to consider the vertex  $v_{i-1}$ . Our plan is try to prove  $w(C_i) \leq wd(v_{i-1})$ . Since in turn we have  $wd(v_{i-1}) \leq wd(v_i)$ , this will prove the lemma.

So, suppose that  $v_{i-1}$  is contained in some subgraph  $G_{j_{i-1}}$ . We confirm that  $G_{j_{i-1}}$  must not be  $G_{j_i}$ , since  $G_{j_i}$  contains only one vertex and  $v_i$  is in  $G_{j_i}$ . Similarly, the discussion on  $G_{j_{i-1}}$  also falls into two cases depending on whether  $G_{j_{i-1}}$  is a graph with a single vertex. In the case where  $G_{j_{i-1}}$  consists of more than one vertices, we will have

$$w(C_i) \leq gc^*(G_{j_{i-1}}) \leq wd_{G_{j_{i-1}}}(v_{i-1}) \leq wd(v_{i-1}) \leq wd(v_i).$$

Otherwise,  $v_{i-1}$  is the only vertex in  $G_{j_{i-1}}$  and we take  $v_{i-2}$  into consideration. Suppose that  $v_{i-2}$  is in subgraph  $G_{j_{i-2}}$  for some  $1 \leq j_{i-2} \leq i$ . We confirm that  $G_{j_{i-2}}$  must be neither  $G_{j_{i-1}}$  nor  $G_{j_i}$  since  $G_{j_{i-1}}$  contains the only vertex  $v_{i-1}$  and  $G_{j_i}$  contains the only vertex  $v_i$ . The proof will recur and we claim that the process must end at some subgraph with more than one vertices, say  $G_{j_\ell}$ . By our notational convention, the vertex in graph  $G_{j_\ell}$  in consideration is  $v_\ell$ . So, at this point we will have

$$w(C_i) \leq gc^*(G_\ell) \leq wd_{G_\ell}(v_\ell) \leq wd(v_\ell) \leq \dots \leq wd(v_{i-1}) \leq wd(v_i),$$

following the lemma.

The only remaining thing is to prove the claim. The proof is by contradiction. So, suppose that when considering the last possible vertex  $v_1$ , we have that the subgraph containing  $v_1$ , which is  $G_{j_1}$  by our convention, still contains only one vertex (i.e.,  $v_1$  itself). So, in this case, we have enumerated  $i$  subgraphs. They are  $G_{j_i}$  containing  $v_i$  only,  $G_{j_{i-1}}$  containing  $v_{i-1}$  only, and so on, and  $G_{j_1}$  containing  $v_1$  only. But if so, then all the  $i$  subgraphs contain  $i$  vertices in total, contradicting the fact that the total vertices in all these  $i$  subgraphs should be  $n > i$ . Note that we have  $i \leq k - 1 \leq n - 2$ . The proof is finished.  $\square$

**Lemma 2.** *Algorithm MG can be implemented in  $O(k(n^2 + \log k))$  time.*

*Proof.* The implementation of Algorithm MG is lightly different to its description. We use a data structure, called minimum heap, to organize the partition  $\mathcal{P}$ . Each part  $S$  of  $\mathcal{P}$  is a node of the heap, with the minimum cut value  $gc^*(G[S])$  being the node's key in the heap. This means that we only need to compute the minimum cut of each part in  $\mathcal{P}$  only once.

The top node in the heap is just the part in  $\mathcal{P}$  with the minimum  $gc^*(\cdot)$  value. So, in each iteration of the algorithm, we remove the top node  $S$  from the heap. Let  $(S', S'')$  be a minimum cut of  $G[S]$ . We then insert  $S'$  and  $S''$  back into the heap. Before doing this, we should have already computed  $gc^*(G[S'])$

and  $gc^*(G[S'''])$ . If a part of  $\mathcal{P}$  is a singleton part (i.e., contains only one vertex), its  $gc^*(\cdot)$  is defined to be infinity.

Computing a minimum cut of a graph with  $n$  vertices takes  $O(n^2)$  time using Stoer and Wagner's algorithm [30]. Each fundamental operation to the minimum heap with  $k$  nodes takes  $O(\log k)$  time. Algorithm MG has  $k - 1$  iterations. Therefore, the algorithm can be implemented in  $O(k(n^2 + \log k))$  time, or roughly speaking,  $O(kn^2)$  time.  $\square$

**Theorem 1.** *Algorithm MG is a  $(1 - \frac{2(k-1)}{n})$ -approximation algorithm for the Max  $k$ -Uncut problem.*

*Proof.* Let  $C_i$  and  $v_i$  be defined as in Sect. 2.2. By Lemma 1, the total weight of all  $C_i$ 's ( $1 \leq i \leq k - 1$ ) satisfies

$$\sum_{i=1}^{k-1} w(C_i) \leq \sum_{i=1}^{k-1} wd(v_i) \leq \frac{k-1}{n} \sum_v wd(v) = \frac{2(k-1)}{n} \sum_e w(e),$$

where the second inequality holds since  $v_1, v_2, \dots, v_{k-1}$  are the first  $k-1$  vertices with the lightest weighted degree (see (1)). Let  $SOL$  be the total weight of happy edges found by Algorithm MG, and  $OPT$  be the optimum value of the Max  $k$ -Uncut problem. So, we have

$$\begin{aligned} SOL &= \sum_e w(e) - \sum_{i=1}^{k-1} w(C_i) \geq \sum_e w(e) - \frac{2(k-1)}{n} \sum_e w(e) \\ &= \left(1 - \frac{2(k-1)}{n}\right) \sum_{e \in E} w(e) \geq \left(1 - \frac{2(k-1)}{n}\right) OPT. \end{aligned}$$

Finally, Lemma 2 says that Algorithm MG runs in polynomial time. The theorem follows.  $\square$

### 3 Experiments

We test Algorithm MG on randomly generated instances. In this section we show the experimental results. For the sake of comparisons, we propose two more algorithms for Max  $k$ -Uncut besides Algorithm MG, namely, the maximum adjacency search algorithm and the binary partition algorithm. We also introduce two existing algorithms for Max  $k$ -Uncut. They are Algorithm R [35] and Algorithm G [35]. These algorithms are used to be compared with Algorithm MG in the experiments.

We test all the five algorithms on 12 data sets, with each data set containing 30,000 instances. The data sets are generated using three graph models, namely, the  $G[n, p]$  model, the configuration model, and the preferential attachment model.

The experimental results on 12 data sets show that Algorithm MG is overwhelmingly better than the other four algorithms tested in the paper. This phenomenon may have some theoretical reasons. First, note that Algorithm MAS,

Algorithm R and Algorithm G all produce big-endian partitions as their solutions. A big-endian partition is obviously a partition with a very special structure. It may be an approximate solution with provable performance guarantee. However, it is hard to imagine that every good solution to Max  $k$ -Uncut is big-endian. In fact, one can easily give an instance of Max  $k$ -Uncut whose optimal solution is not big-endian. In contrast, Algorithm MG outputs a more natural partition, which is usually not big-endian.

Second, in the experiments, Algorithm MG and Algorithm BP behave better than the other three algorithms. This shows that computing the minimum cuts may be one of the right ways to attack the Max  $k$ -Uncut problem. Moreover, Algorithm MG is still better than Algorithm BP according to the experimental results. Algorithm MG is somewhat smart, since it always cuts on the weakest part of the current subgraphs. In contrast, Algorithm BP is somewhat awkward. It always cuts the subgraph in the forefront.

It is also interesting to note that Algorithm MG and Algorithm G have the same approximation ratio  $1 - \frac{2(k-1)}{n}$ . However, their practical performances in the experiments are contrasting distinctly. This is mainly because approximation ratio analysis is a worst-case analysis. Obviously, it is hard to reflect the general performance of an algorithm only by worst-case analysis. We may need more theory to comprehensively evaluate an algorithm. Experiment is a complementary means to reveal more details of an algorithm.

Due to space limitation, the four more tested algorithms (i.e., Algorithms MAS, BP, R and G), the data sets, and the detailed experimental results will be given in the full version of the paper.

## 4 Conclusions

The Max  $k$ -Uncut problem arises from the study of network science. This problem is the complement of the classic Min  $k$ -Cut problem, and is strongly related to the famous Densest  $k$ -Subgraph problem. We propose a combinatorial algorithm, namely, Algorithm MG, for the Max  $k$ -Uncut problem with approximation ratio  $1 - \frac{2(k-1)}{n}$ . Algorithm MG is very simple in the sense that it consists of only  $k - 1$  min cut computations. The time complexity of Algorithm MG is  $O(kn^2)$ , implying that it is a practical algorithm. The experimental results show that Algorithm MG has high quality performance in the tested graph models, including the  $G[n, p]$  model, the configuration model, and the preferential attachment model.

**Acknowledgements.** This work is supported by the National Natural Science Foundation of China (61972228 and 61672323), and the Natural Science Foundation of Shandong Province (ZR2021ZD15 and ZR2019MF072).



## References

1. Agarwal, A., Charikar, M., Makarychev, K., Makarychev, Y.:  $O(\sqrt{\log n})$  approximation algorithms for min uncut, min 2CNF deletion, and directed cut problems. In: Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC), pp. 573–581 (2005)
2. 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 the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 61–72 (2017)
3. Bhaskara, A., Charikar, M., Chlamtac, E., Feige, U., Vijayaraghavan, A.: Detecting high log-densities: an  $O(n^{1/4})$  approximation for densest  $k$ -subgraph. In: Proceedings of the 42nd Annual ACM Symposium on Theory of Computing (STOC), pp. 201–210 (2010)
4. Bliznets, I., Sagunov, D.: On happy colorings, cuts, and structural parameterizations. In: Proceedings of the 45th International Workshop of Graph-Theoretic Concepts in Computer Science (WG), pp. 148–161 (2019)
5. Bliznets, I., Sagunov, D.: Lower bounds for the happy coloring problems. Theoret. Comput. Sci. **838**, 94–110 (2020)
6. Brandes, U., et al.: On modularity clustering. IEEE Trans. Knowl. Data Eng. **20**(2), 172–188 (2008)
7. Choudhurya, S., Gaurb, D.R., Krishnamurtic, R.: An approximation algorithm for max  $k$ -uncut with capacity constraints. Optimization **61**(2), 143–150 (2012)
8. Dasgupta, S.: A cost function for similarity-based hierarchical clustering. In: Proceedings of the 48th Annual ACM Symposium on Theory of Computing (STOC), pp. 118–127 (2016)
9. Freeman, L.: A set of measures of centrality based on betweenness. Sociometry **40**(1), 35–41 (1977)
10. Frieze, A., Jerrum, M.: Improved approximation algorithms for max  $k$ -cut and max bisection. Algorithmica **18**, 67–81 (1997)
11. Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. Proc. Natl. Acad. Sci. U.S.A. **99**(12), 7821–7826 (2002)
12. Goemans, M.X., Williamson, D.P.: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. J. ACM **42**(6), 1115–1145 (1995)
13. Goldschmidt, O., Hochbaum, D.: A polynomial algorithm for the  $k$ -cut problem for fixed  $k$ . Math. Oper. Res. **19**(1), 24–37 (1994)
14. Hochbaum, D., Shmoys, D.: A best possible heuristic for the  $k$ -center problem. Math. Oper. Res. **10**(2), 180–184 (1985)
15. Jain, K., Vazirani, 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)
16. 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. In: Proceedings of the 18th Annual ACM Symposium on Computational Geometry (SoCG), pp. 10–18 (2002)
17. Langberg, M., Rabani, Y., Swamy, C.: Approximation algorithms for graph homomorphism problems. In: Proceedings of the 9th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX), pp. 176–187 (2006)

18. Lewis, R., Thiruvady, D., Morgan, K.: Finding happiness: an analysis of the maximum happy vertices problem. *Comput. Oper. Res.* **103**, 265–276 (2019)
19. Lewis, R., Thiruvady, D.R., Morgan, K.: The maximum happy induced subgraph problem: bounds and algorithms. *Comput. Oper. Res.* **126**, 105114:1–105114:15 (2021)
20. Li, A., Pan, Y.: Structural information and dynamical complexity of networks. *IEEE Trans. Inf. Theory* **62**(6), 3290–3339 (2016)
21. Li, A., et al.: Decoding topologically associating domains with ultra-low resolution hi-c data by graph structural entropy. *Nat. Commun.* **9**, 3265:1–3265:12 (2018)
22. Lloyd, S.: Least squares quantization in PCM. *IEEE Trans. Inf. Theory* **28**(2), 129–137 (1982)
23. Newman, M.E.J.: From the cover: the structure of scientific collaboration networks. *Proc. Natl. Acad. Sci. U.S.A.* **98**(2), 404–409 (2001)
24. Newman, M.E.J.: Modularity and community structure in networks. *Proc. Natl. Acad. Sci. U.S.A.* **103**(23), 8577–8582 (2006)
25. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. *Phys. Rev. E* **69**, 026113:1–026113:15 (2004)
26. Parchas, P., Naamad, Y., Van Bouwel, P., Faloutsos, C., Petropoulos, M.: Fast and effective distribution-key recommendation for amazon redshift. *Proc. VLDB Endow.* **13**(11), 2411–2423 (2020)
27. Raghavan, U.N., Albert, R., Kumara, S.: Near linear-time algorithm to detect community structures in large-scale networks. *Phys. Rev. E* **76**(3), 036106 (2007)
28. Ravi, S.S., Rosenkrantz, D., Tayi, G.: Heuristic and special case algorithms for dispersion problems. *Oper. Res.* **42**, 299–310 (1994)
29. Saran, H., Vazirani, V.: Finding  $k$ -cuts within twice the optimal. *SIAM J. Comput.* **24**, 101–108 (1995)
30. Stoer, M., Wagner, F.: A simple min-cut algorithm. *J. ACM* **44**(4), 585–591 (1997)
31. Wu, C., Xu, D., Du, D., Xu, W.: A complex semidefinite programming rounding approximation algorithm for the balanced max-3-uncut problem. In: Cai, Z., Zelikovsky, A., Bourgeois, A. (eds.) *COCOON 2014*. LNCS, vol. 8591, pp. 324–335. Springer, Cham (2014). [https://doi.org/10.1007/978-3-319-08783-2\\_28](https://doi.org/10.1007/978-3-319-08783-2_28)
32. Ye, Y., Zhang, J.: Approximation of dense- $n/2$ -subgraph and the complement of min-bisection. *J. Global Optim.* **25**(1), 55–73 (2003)
33. Zhang, P., Li, A.: Algorithmic aspects of homophily of networks. *Theoret. Comput. Sci.* **593**, 117–131 (2015)
34. Zhang, P., Liu, Z.: Approximating max  $k$ -uncut via LP-rounding plus greed, with applications to densest  $k$ -subgraph. *Theoret. Comput. Sci.* **849**, 173–183 (2021)
35. Zhang, P., Chenchen, W., Dachuan, X.: Approximation and hardness results for the max  $k$ -uncut problem. *Theoret. Comput. Sci.* **749**, 47–58 (2018)
36. Zhang, P., Yao, X., Jiang, T., Li, A., Lin, G., Miyano, E.: Improved approximation algorithms for the maximum happy vertices and edges problems. *Algorithmica* **80**, 1412–1438 (2018)



# On Two Types of Concept Lattices in the Theory of Numberings

Nikolay Bazhenov<sup>1</sup> , Manat Mustafa<sup>2</sup> , and Anvar Nurakunov<sup>3</sup>

<sup>1</sup> Sobolev Institute of Mathematics, 4 Acad. Koptyug Ave.,  
Novosibirsk 630090, Russia  
[bazhenov@math.nsc.ru](mailto:bazhenov@math.nsc.ru)

<sup>2</sup> Department of Mathematics, School of Sciences and Humanities,  
Nazarbayev University, 53 Qabanbaybatyr Avenue, Astana 010000, Kazakhstan  
[manat.mustafa@nu.edu.kz](mailto:manat.mustafa@nu.edu.kz)

<sup>3</sup> Institute of Mathematics of the National Academy of Sciences, 265a Chui prosp.,  
Bishkek 720071, Kyrgyzstan

**Abstract.** The theory of numberings studies uniform computations for families of mathematical objects. A large body of literature is devoted to investigations of Rogers semilattices for computable families  $S$ , i.e. uniformly enumerable families of computably enumerable subsets of the set of natural numbers  $\omega$ . Working within the framework of Formal Concept Analysis, we introduce two approaches to classification of at most countable families  $S \subset P(\omega)$ . Similarly to the classical theory of numberings, each of the approaches assigns to a family  $S$  its own concept lattice. Our first approach captures the cardinality of a family  $S$ . We prove the following: if  $S$  contains at least two elements, then the corresponding concept lattice is a modular lattice of height 3 such that the number of its atoms equals the cardinality of  $S$ . Our second approach provides a much richer environment: we show that any countable complete lattice can be obtained as the concept lattice induced by an appropriate family  $S$ .

In addition, we employ the index sets technique, and consider the following isomorphism problem: given two computable families  $S$  and  $T$ , how hard is it to determine whether the corresponding concept lattices are isomorphic? The isomorphism problem for the first approach is a  $\Pi_3^0$ -complete set, and the isomorphism problem for the second approach is  $\Sigma_1^1$ -hard.

**Keywords:** Numbering · Index set · Computable structure · Concept lattice · Formal Concept Analysis

---

The work was supported by Nazarbayev University Faculty Development Competitive Research Grants N021220FD3851. The work of Bazhenov is supported by the Mathematical Center in Akademgorodok under the agreement No. 075-15-2022-281 with the Ministry of Science and Higher Education of the Russian Federation.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022  
D.-Z. Du et al. (Eds.): TAMC 2022, LNCS 13571, pp. 79–92, 2022.  
[https://doi.org/10.1007/978-3-031-20350-3\\_8](https://doi.org/10.1007/978-3-031-20350-3_8)

# 1 Introduction

The theory of numberings investigates uniform computational procedures for families of mathematical objects. Let  $\mathcal{S}$  be an at most countable family. A *numbering*  $\nu$  of the family  $\mathcal{S}$  is a surjective map from the set of natural numbers  $\omega$  onto  $\mathcal{S}$ .

Numberings have emerged as an important methodological tool with the rise of the modern formal notion of an algorithmic computation. Gödel [13] employed an effective numbering of first-order formulae in the proof of his seminal incompleteness theorems. Kleene [17] constructed the celebrated numbering of the family of all partial recursive functions—this is a list  $\{\varphi_e(x)\}_{e \in \omega}$  enumerating all unary partial recursive functions. The key property of the numbering is that the binary function  $\psi(e, x) := \varphi_e(x)$  is also partial recursive. In the 1950s, the foundations of the modern theory of numberings were developed by Kolmogorov and Uspenskii [18, 22] and, independently, by Rogers [20].

The reader is referred to the monograph [21] for the background on computability theory. Since the 1960s, numberings for families of computably enumerable (c.e.) sets have been intensively studied by computability theorists. Let  $\mathcal{S}$  be a family of c.e. sets. A numbering  $\nu$  of  $\mathcal{S}$  is *computable* if the set  $\{(k, x) : k \in \omega, x \in \nu(k)\}$  is computably enumerable. Informally, a computable numbering provides a uniform algorithmic enumeration for the family  $\mathcal{S}$ . A family  $\mathcal{S}$  is *computable* if it admits a computable numbering.

A measure of relative complexity for numberings is provided by the notion of *reducibility*. A numbering  $\nu$  is *reducible* to a numbering  $\mu$ , denoted by  $\nu \leq \mu$ , if there is a total computable function  $f(x)$  such that  $\nu(k) = \mu(f(k))$  for all  $k \in \omega$ . Informally, a reduction  $\nu \leq \mu$  is realized by an algorithmic procedure, which, given a  $\nu$ -index of an object  $a \in \mathcal{S}$ , outputs a  $\mu$ -index of  $a$ .

Let  $\mathcal{S}$  be a computable family of c.e. sets. In a standard recursion-theoretic way, the reducibility  $\leq$  induces an upper semilattice  $\mathcal{R}(\mathcal{S})$ —this semilattice contains the  $\leq$ -degrees of all computable numberings of  $\mathcal{S}$ . The structure  $\mathcal{R}(\mathcal{S})$  is called the *Rogers semilattice* of the family  $\mathcal{S}$ .

Rogers semilattices provide an important classification tool, which allows one to compare algorithmic properties of different computable families. We should note that in general, studying isomorphism types of Rogers semilattices is notoriously hard. We also note that there is a fruitful line of research, which studies Rogers semilattices for *generalized* computable numberings—in particular, for numberings in various recursion-theoretic hierarchies. We refer the reader, e.g., to [2, 6–9, 15] for the research on Rogers semilattices.

In this paper, we introduce two approaches to classification of families of sets  $\mathcal{S}$  and their numberings. These approaches are based on Formal Concept Analysis [24]. Similarly to the notion of Rogers semilattice, for an at most countable family  $\mathcal{S}$ , we define the lattices  $\mathbf{FC}_i(\mathcal{S})$ , where  $i \in \{1, 2\}$ .

The paper is arranged as follows. Section 2 provides the necessary preliminaries. In Sect. 3, we discuss the first approach, which works with concept lattices  $\mathbf{FC}_1(\mathcal{S})$  (see Definition 3.1). For a family  $\mathcal{S}$ , concepts  $\Delta$  from  $\mathbf{FC}_1(\mathcal{S})$  have the

following property: the extent of  $\Delta$  is a subset of  $\omega$ , and the intent of  $\Delta$  is a subfamily of  $\mathcal{S}$ .

In Theorem 3.1, we obtain a complete description of all possible isomorphism types of  $\mathbf{FC}_1(\mathcal{S})$ . If  $\mathcal{S}$  has only one element, then  $\mathbf{FC}_1(\mathcal{S})$  is a one-element lattice; otherwise,  $\mathbf{FC}_1(\mathcal{S})$  is a modular lattice of height 3 with precisely  $\kappa$  atoms, where  $\kappa$  is the cardinality of  $\mathcal{S}$ . Working within the framework of index sets [5, 14], Subsect. 3.2 establishes algorithmic complexity of the following *isomorphism problem*: for two computable families  $\mathcal{S}$  and  $\mathcal{T}$ , when the corresponding lattices  $\mathbf{FC}_1(\mathcal{S})$  and  $\mathbf{FC}_1(\mathcal{T})$  are isomorphic? Theorem 3.3 proves that the index set associated with this problem is many-one complete in the class  $\Pi_3^0$  of the arithmetical hierarchy.

Section 4 discusses our second approach, which works with concept lattices  $\mathbf{FC}_2(\mathcal{S})$  (Definition 4.1). We show that every countable complete lattice can be realized as  $\mathbf{FC}_2(\mathcal{S})$  for an appropriately chosen family  $\mathcal{S}$  (Theorem 4.1). Furthermore, the real unit interval  $[0; 1]$  and the Boolean lattice  $\mathcal{P}(\omega)$  of all subsets of  $\omega$  can be realized as  $\mathbf{FC}_2(\mathcal{S})$  (Subsect. 4.1). Theorem 4.2 proves that any set from the class  $\Sigma_1^1$  of the analytical hierarchy is many-one reducible to the index set, which encodes the isomorphism problem for lattices  $\mathbf{FC}_2(\mathcal{S})$ . Section 5 contains further discussion.

## 2 Preliminaries

Following the usual conventions of computability theory, by  $\omega$  we denote the set of natural numbers. The set of all subsets of  $\omega$  is denoted by  $\mathcal{P}(\omega)$ . For a set  $X$ , by  $\text{card}(X)$  we denote the cardinality of  $X$ . If  $R \subseteq A \times B$  is a binary relation, then we use notations:

$$\pi_A(R) = \{a \in A : \exists b[(a, b) \in R]\}, \quad \pi_B(R) = \{b \in B : \exists a[(a, b) \in R]\}.$$

For a pair of natural numbers  $(k, \ell)$ , by  $\langle k, \ell \rangle$  we denote its standard Cantor index, i.e.

$$\langle k, \ell \rangle = \frac{(k + \ell)(k + \ell + 1)}{2} + k.$$

We assume that the reader is familiar with the basic notions of computability theory and computable structure theory. We refer to the monographs [1, 21] for the background.

The preliminaries on Formal Concept Analysis follow the monograph [12].

Recall that a *formal context*  $\mathbb{K} = (G, M, I)$  consists of the set of objects  $G$ , the set of attributes  $M$ , and the incidence relation  $I \subseteq G \times M$ . If  $\mathbb{K}$  is a formal context and  $A \subseteq G$ , then

$$\alpha_{\mathbb{K}}(A) := \{m \in M : (\forall g \in A)[(g, m) \in I]\}.$$

For  $B \subseteq M$ , set

$$\beta_{\mathbb{K}}(B) := \{g \in G : (\forall m \in B)[(g, m) \in I]\}.$$

If the triple  $\mathbb{K}$  is clear from the discussion, then we omit the subscript  $\mathbb{K}$ —e.g., we write  $\alpha(A)$  in place of  $\alpha_{\mathbb{K}}(A)$ .

A *formal concept* of the context  $\mathbb{K}$  is a pair  $(A, B)$  such that  $A \subseteq G$ ,  $B \subseteq M$ ,  $B = \alpha_{\mathbb{K}}(A)$ , and  $A = \beta_{\mathbb{K}}(B)$ . For a formal context  $\Delta = (A, B)$ ,  $A$  is called the *extent* of  $\Delta$ , and  $B$  is the *intent* of  $\Delta$ .

The ordering  $\leq$  of the concepts of  $\mathbb{K}$  is defined as follows:

$$(A_0, B_0) \leq (A_1, B_1) \Leftrightarrow A_0 \subseteq A_1 \Leftrightarrow B_0 \supseteq B_1.$$

The Basic Theorem on Concept Lattices (see [12]) establishes the following:

1. The ordering  $\leq$  on the set of all concepts of  $\mathbb{K}$  induces a complete lattice. This lattice is called the *concept lattice* of  $\mathbb{K}$ , and we denote it by  $\mathbb{L}(\mathbb{K})$ .
2. Let  $\mathcal{L}$  be a complete lattice. Consider the formal context  $\mathbb{K}_{\mathcal{L}} = (\mathcal{L}, \mathcal{L}, \leq_{\mathcal{L}})$ . Then the lattice  $\mathbb{L}(\mathbb{K}_{\mathcal{L}})$  is isomorphic to  $\mathcal{L}$ . In addition, every concept of  $\mathbb{K}_{\mathcal{L}}$  is of the form

$$(\hat{a}, \check{a}) = (\{b : b \leq_{\mathcal{L}} a\}, \{c : a \leq_{\mathcal{L}} c\}) \quad (1)$$

for some element  $a \in \mathcal{L}$ .

### 3 The First Approach: Capturing the Cardinality of a Family

Our first approach is based on the following definition:

**Definition 3.1.** *Let  $\mathcal{S}$  be an at most countable family, and let  $\nu$  be a numbering of the family  $\mathcal{S}$ . Consider the relation*

$$I_{\nu} := \{(n, \nu(n)) : n \in \omega\}.$$

By  $\mathbf{FC}_1(\mathcal{S})$  we denote the concept lattice of the formal context  $\mathbb{K} = (\omega, \mathcal{S}, I_{\nu})$ .

It turns out that, informally speaking, the isomorphism type of the lattice  $\mathbf{FC}_1(\mathcal{S})$  encodes *only the cardinality* of the family  $\mathcal{S}$ . This is witnessed by the following result.

For a natural number  $n \geq 2$ , let  $M_n$  be a modular lattice of height 3 with  $n$  atoms. By  $M_{\omega}$  we denote a modular lattice of height 3 with countably many atoms.

**Theorem 3.1.** *Let  $\mathcal{S}$  be an at most countable, non-empty family. If  $\mathcal{S}$  contains only one element, then  $\mathbf{FC}_1(\mathcal{S})$  is a one-element lattice. Otherwise,  $\mathbf{FC}_1(\mathcal{S})$  is isomorphic to  $M_{\text{card}(\mathcal{S})}$ .*

Note that Theorem 3.1 justifies our choice of notations in Definition 3.1—we will show that the isomorphism type of the structure  $\mathbf{FC}_1(\mathcal{S})$  *does not depend* on the choice of a numbering  $\nu$ . Informally speaking, the lattice  $\mathbf{FC}_1(\mathcal{S})$  is an invariant, which provides some kind of characterization for all possible numberings of  $\mathcal{S}$ .

The rest of the section is arranged as follows. Subsection 3.1 proves Theorem 3.1. In Subsect. 3.2, we apply the technique of index sets (to be elaborated) to answer the following question:

**Problem A.** How hard is it to determine whether for given computable families  $\mathcal{S}_0$  and  $\mathcal{S}_1$ , the lattices  $\mathbf{FC}_1(\mathcal{S}_0)$  and  $\mathbf{FC}_1(\mathcal{S}_1)$  are isomorphic?

**3.1 Proof of Theorem 3.1**

Let  $\nu$  be an arbitrary numbering of the family  $\mathcal{S}$ .

First, note the following. If the family  $\mathcal{S}$  contains only one element  $m_0$ , then  $\nu(k) = m_0$  for all  $k \in \omega$ . This implies that the formal context  $(\omega, \mathcal{S}, I_\nu)$  has only one concept  $(\omega, \{m_0\})$ . So, in what follows we assume that  $\text{card}(\mathcal{S}) \geq 2$ .

In order to prove the theorem, we obtain a more general algebraic fact:

**Theorem 3.2.** *Let  $\mathbb{K} = (G, N, I)$  be a formal context such that  $\text{card}(N) \geq 2$ . Let  $H$  be the least closed subset of  $G$ , i.e.*

$$H = \{x \in G : (\forall y \in N)[(x, y) \in I]\}.$$

(a) *Suppose that the context  $\mathbb{K}$  satisfies the following condition:*

( $\star$ ) *There is a subset  $G_1$  of  $G$  such that:*

*\*  $G_1 \cap H = \emptyset$ ,*

*\*  $I \subseteq (G_1 \cup H) \times N$ , and*

*\* the set  $I_1 := I \setminus (H \times N)$  is the graph of a surjective map from  $G_1$  onto  $N$ .*

*Then the concept lattice  $\mathbb{L}(\mathbb{K})$  is isomorphic to  $M_{\text{card}(N)}$ .*

(b) *Furthermore, if the set  $N$  is finite and  $\mathbb{L}(\mathbb{K}) \cong M_{\text{card}(N)}$ , then  $\mathbb{K}$  satisfies ( $\star$ ).*

Notice the following. One can apply item (a) of Theorem 3.2 to our context  $(\omega, \mathcal{S}, I_\nu)$ : here we have  $H = \emptyset$ ,  $G_1 = \omega$ , and  $I_1 = I_\nu$ . Therefore, we immediately obtain that the lattice  $\mathbf{FC}_1(\mathcal{S})$  is isomorphic to  $M_{\text{card}(\mathcal{S})}$ .

**Remark 3.1.** In general, one cannot omit the assumption of finiteness of  $N$  in item (b) of Theorem 3.2. This is witnessed by the following context  $(G, N, I)$ : define  $G = N = \omega$  and  $I = \{(x, x + 1) : x \in \omega\}$ .

*Proof (of Theorem 3.2).* (a) Suppose that  $I_1$  is the graph of a surjective map  $\psi$  from some  $G_1 \subseteq G$  onto  $N$ .

Consider an arbitrary set  $A \subseteq G$ . The set  $A$  satisfies one of the following four cases:

*Case 1.* Suppose that  $A \subseteq H$ . Then we have  $\alpha(A) = N$  and  $\beta \circ \alpha(A) = H$ .

*Case 2.* Suppose that  $A \not\subseteq G_1 \cup H$ . Then by considering an element  $k \in A \setminus (G_1 \cup H)$ , one can deduce that  $\alpha(A) = \emptyset$ , and the  $\beta \circ \alpha$ -closure of  $A$  equals  $G$ .

*Case 3.* Suppose that  $A \subseteq G_1 \cup H$ , and there are  $k, \ell \in A \cap G_1$  with  $\psi(k) \neq \psi(\ell)$ . Then we have  $(k, \psi(k)) \in I$  and  $(k, b) \notin I$  for all  $b \neq \psi(k)$ . Therefore,

$$\alpha(A) = \{y \in N : (\forall x \in A)[(x, y) \in I]\} = \emptyset \text{ and } \beta \circ \alpha(A) = G.$$

*Case 4.* Otherwise,  $A \subseteq G_1 \cup H$ ,  $A \cap G_1 \neq \emptyset$ , and for all  $k, \ell \in A \cap G_1$ , we have  $\psi(k) = \psi(\ell)$ . Fix an element  $k_0 \in A \cap G_1$ . Then

$$\alpha(A) = \{\psi(k_0)\} \text{ and } \beta \circ \alpha(A) = H \cup \{\ell \in G_1 : \psi(\ell) = \psi(k_0)\}.$$

The described cases show that every  $\beta\alpha$ -closed set  $A \notin \{H, G\}$  has the form

$$A_b = H \cup \{\ell \in G_1 : \psi(\ell) = b\} \text{ for some } b \in N.$$

Clearly, if  $b \neq b'$ , then  $A_b$  and  $A_{b'}$  are incomparable under  $\subseteq$ . Hence, we deduce that the concept lattice  $\mathbb{L}(\mathbb{K})$  is isomorphic to  $M_{\text{card}(N)}$ .

(b) First, we obtain an auxiliary result on subdirect relations.

**Definition 3.2.** A binary relation  $R \subseteq A \times B$  is subdirectly  $A$ -proper if  $\pi_A(R) = A$ ,  $\pi_B(R) = B$ , and there is no  $b \in B$  such that  $A \times \{b\} \subseteq R$ .

In other words, a relation  $R$  is subdirectly  $A$ -proper if it satisfies the following conditions:

- (i) for every  $a \in A$ , there is  $b \in B$  such that  $(a, b) \in R$ ;
- (ii) for every  $b \in B$ , the set  $\{a \in A : (a, b) \in R\}$  is a non-empty proper subset of  $A$ .

In particular, if  $\text{card}(B) \geq 2$ , then for any surjective map  $f: A \rightarrow B$ , its graph  $R_f = \{(a, f(a)) : a \in A\}$  is a subdirectly  $A$ -proper relation.

**Proposition 3.1.** Let  $(A, B, R)$  be a formal context such that  $2 \leq \text{card}(B) < \omega$ . Suppose that the concept lattice  $\mathbb{L}(A, B, R)$  is isomorphic to  $M_{\text{card}(B)}$ . Then there is a subset  $A^0$  of  $A$  such that  $R \subseteq A^0 \times B$  and  $R$  is subdirectly  $A^0$ -proper.

*Proof.* Let

$$A^0 = \{a \in A : (\exists b \in B)((a, b) \in R)\}.$$

It is easy to see that  $R \subseteq A^0 \times B$  and  $\pi_{A^0}(R) = A^0$ . Moreover, one can check that  $\mathbb{L}(A^0, B, R) \cong \mathbb{L}(A, B, R)$ .

Since the lattice of the  $\beta\alpha$ -closed subsets of  $A^0$  is isomorphic to  $M_n$ , where  $n = \text{card}(B)$ , there are  $\beta\alpha$ -closed subsets  $A_i$ ,  $1 \leq i \leq n$ , of the set  $A^0$  such that  $A_i \neq A_j$  and  $A_i \vee A_j = A_i \vee A_k = A^0$  for all  $j \neq i \neq k$ . It means  $\alpha(A_i) \cap \alpha(A_j) = \alpha(A_i) \cap \alpha(A_k)$ . In addition,  $\alpha(A_i) \neq \alpha(A_j)$  for all  $i \neq j$  because  $A_i$  and  $A_j$  are different closed subsets.

By definition,  $A_i \subseteq \beta(\{b\})$  for every  $b \in \alpha(A_i) \setminus (\alpha(A_i) \cap \alpha(A_j))$ . Assume that  $A_i$  is a strict subset of  $\beta(\{b\})$  for each  $b \in \alpha(A_i) \setminus (\alpha(A_i) \cap \alpha(A_j))$ . This means  $\beta(\{b\}) = A^0$  for all  $b \in \alpha(A_i) \setminus (\alpha(A_i) \cap \alpha(A_j))$ . Thus,

$$\beta(\alpha(A_i) \setminus (\alpha(A_i) \cap \alpha(A_j))) = \bigcap \{\beta(\{b\}) : b \in \alpha(A_i) \setminus (\alpha(A_i) \cap \alpha(A_j))\} = A^0.$$



Hence,

$$\begin{aligned} A_i &= \beta(\alpha(A_i)) = \beta((\alpha(A_i) \setminus (\alpha(A_i) \cap \alpha(A_j))) \cup (\alpha(A_i) \cap \alpha(A_j))) = \\ &= \beta(\alpha(A_i) \setminus (\alpha(A_i) \cap \alpha(A_j))) \cap \beta(\alpha(A_i) \cap \alpha(A_j)) = A^0. \end{aligned}$$

This is impossible by the choice of  $A_i$ . Therefore, for every  $i \leq n$ , there is an element  $b_i \in \alpha(A_i)$  such that  $\beta(\{b_i\}) = A_i$ .

Note that  $\beta(\{b_i\}) \neq \beta(\{b_j\})$  for all  $i \neq j$ . Since  $\text{card}(B) = n$ , we get  $B = \{b_i : 1 \leq i \leq n\}$ . Hence,  $\pi_B(R) = B$ . Summarizing, we showed that  $\pi_{A^0}(R) = A^0$ ,  $\pi_B(R) = B$ , and  $\beta(\{b_i\}) = A_i \subsetneq A^0$ . That is,  $R$  is subdirectly  $A^0$ -proper. Proposition 3.1 is proved.  $\square$

Now consider the context  $\mathbb{K} = (G, N, I)$  from item (b) of Theorem 3.2. Let  $n = \text{card}(N)$ . We apply the proof of Proposition 3.1 to the context  $\mathbb{K}$ , and obtain that  $N = \{b_i : 1 \leq i \leq n\}$  and  $A_i = \beta(\{b_i\})$ .

It is clear that  $A_i \cap A_j = A_i \cap A_k = H$  for all  $j \neq i \neq k$ . We put

$$G_1 := \left( \bigcup \{A_i : i \leq n\} \right) \setminus H.$$

It is not hard to check that  $I_1 := I \setminus (H \times N)$  is the graph of a map  $f : G_1 \rightarrow N$  defined by  $f(a) = b_i$  for any  $a \in A_i \cap G_1$  and  $i \leq n$ . Therefore, we deduce that the context  $\mathbb{K}$  satisfies the condition  $(\star)$ .

This concludes the proofs of Theorem 3.2 and Theorem 3.1.  $\square$

### 3.2 Classification via Index Sets

Index sets provide an important classification tool in the field of computable structure theory. As an example of a recent application of index sets, we mention the following: the paper [4] established that there is no reasonable syntactic characterization of the algebraic structures with a polynomial-time presentation. For a detailed discussion of index sets for computable structures, we refer to the surveys [5, 10].

Here we work with index sets in the setting of the theory of numberings. Within this framework, the systematic investigations of index sets were initiated by McLaughlin [19]. For the known results in this area, the reader is referred to [23].

Recall that  $\{\varphi_e\}_{e \in \omega}$  is Kleene's numbering of the family of all unary partial computable functions. As usual, for  $e \in \omega$ ,  $W_e$  denotes the c.e. set  $\text{dom}(\varphi_e)$ .

We consider the following effective listing. For  $i, k \in \omega$ , let

$$\theta_i(k) = \begin{cases} W_{\varphi_i(k)}, & \text{if } \varphi_i(k) \text{ is defined,} \\ \emptyset, & \text{otherwise.} \end{cases}$$

It is well-known that the list  $\{\theta_i\}_{i \in \omega}$  enumerates *all* computable numberings of *all* computable families. In addition, there exists a total computable function  $h_0(x, y)$  such that  $\theta_i(k) = W_{h_0(i, k)}$  for all  $i$  and  $k$ .

For  $i \in \omega$ , by  $\mathcal{T}_i$  we denote the family of c.e. sets, which is indexed by the numbering  $\theta_i$ , i.e.:

$$\mathcal{T}_i = \{\theta_i(k) : k \in \omega\}.$$

The described framework allows us to re-cast Problem A in a more formal setting:

**Problem A Revisited.** Find the algorithmic complexity of the set

$$\mathbf{Iso}_1 := \{(i, j) \in \omega \times \omega : \mathbf{FC}_1(\mathcal{T}_i) \cong \mathbf{FC}_1(\mathcal{T}_j)\}.$$

The following theorem provides a solution to this problem—the set  $\mathbf{Iso}_1$  finds its exact place in the arithmetical hierarchy.

**Theorem 3.3.** *The set  $\mathbf{Iso}_1$  is  $\Pi_3^0$ -complete.*

For reasons of space, the proof of Theorem 3.3 is given in Appendix A.

## 4 The Second Approach: Realizing All Countable Complete Lattices

In this section, we work only with at most countable, non-empty families  $\mathcal{S} \subset P(\omega)$ . Our second approach is based on the following definition:

**Definition 4.1.** *Let  $\nu$  be a numbering of a family  $\mathcal{S}$ . Consider a binary relation*

$$Q_\nu = \{(x, n) : n \in \omega, x \in \nu(n)\} \subseteq \omega \times \omega.$$

By  $\mathbf{FC}_2(\mathcal{S})$  we denote the concept lattice  $\mathbb{L}(\omega, \omega, Q_\nu)$ .

The second approach provides a much richer environment. This is witnessed by the following result:

**Theorem 4.1.(a)** *Let  $\mathcal{S}$  be a family of subsets of  $\omega$ . The structure  $\mathbf{FC}_2(\mathcal{S})$  is well-defined, i.e. the isomorphism type of the lattice  $\mathbf{FC}_2(\mathcal{S})$  does not depend on the choice of a numbering  $\nu$ .*

**(b)** *Let  $\mathcal{L}$  be a countable complete lattice. Then there is a family  $\mathcal{S}_\mathcal{L}$  such that  $\mathbf{FC}_2(\mathcal{S}_\mathcal{L}) \cong \mathcal{L}$ .*

For reasons of space, the Proof of Theorem 4.1 is given in Appendix B. In what follows, we work within the framework of Subsect. 3.2, and find the complexity of index set

$$\mathbf{Iso}_2 := \{(i, j) \in \omega \times \omega : \mathbf{FC}_2(\mathcal{T}_i) \cong \mathbf{FC}_2(\mathcal{T}_j)\}. \quad (2)$$

After that, in Subsect. 4.1, we provide examples of *computable* families  $\mathcal{S}$  with *uncountable* lattices  $\mathbf{FC}_2(\mathcal{S})$ .

First, we note the following observation. The proof of item (b) of Theorem 4.1 is effective in the following sense: Given a computable partial order  $\mathcal{L} = (\omega; \leq_\mathcal{L})$ , which is a complete lattice, one can effectively produce a *computable* numbering

$$\nu_\mathcal{L}(n) := \{k \in \omega : k \leq_\mathcal{L} n\} \quad (3)$$

such that the numbering  $\nu_{\mathcal{L}}$  indexes a family  $\mathcal{S}_2[\mathcal{L}]$  of c.e. sets, and the lattice  $\mathbf{FC}_2(\mathcal{S}_2[\mathcal{L}])$  is isomorphic to  $\mathcal{L}$ .

This observation plays a key role in our next result. Informally speaking, this result shows that in contrast to our first approach (see Theorem 3.3), a similar classification problem for the second approach is significantly harder: every set from the level  $\Sigma_1^1$  of the analytical hierarchy is many-one reducible to the index set  $\mathbf{Iso}_2$  from (2).

**Theorem 4.2.** *The set  $\mathbf{Iso}_2$  is  $\Sigma_1^1$ -hard.*

*Proof.* It is well-known that the isomorphism problem  $E(UG)$  for computable undirected graphs is a  $\Sigma_1^1$ -complete set. The proof of this fact can be obtained, e.g., by combining the proof sketch, given by Theorem 3.1 of [5], with the methods of Sect. 3.1 of [16]. As a by-product of the proof, one can easily get the following useful fact.

Let  $X \subseteq \omega$  be an arbitrary  $\Sigma_1^1$  set such that  $0 \in X$ . Then there exists a computable sequence  $(\mathcal{G}_n)_{n \in \omega}$  of computable undirected graphs such that for every  $n \in \omega$ ,

- $\text{dom}(\mathcal{G}_n) = \omega$ , and for each  $i > 0$ , there is an edge between 0 and  $i$  in  $\mathcal{G}_n$ .
- we have  $n \in X$  if and only if  $\mathcal{G}_n \cong \mathcal{G}_0$ .

We fix such a sequence  $(\mathcal{G}_n)_{n \in \omega}$ . Given  $n \in \omega$ , we build a computable partial order  $\mathcal{A}_n$ . Essentially, this construction is a simplified version of that provided by Sect. 3.3 of [16].

- The domain  $\text{dom}(\mathcal{A}_n)$  consists of the following parts:
  - the  $\leq_{\mathcal{A}_n}$ -least element  $\perp$  and the  $\leq_{\mathcal{A}_n}$ -greatest element  $\top$ ;
  - the set  $A = \{a_i : i \in \omega\}$ ;
  - the set  $D_n = \{d_{i,j} : i < j, \text{ and the graph } \mathcal{G}_n \text{ has edge between } i \text{ and } j\}$ .
- If  $\mathcal{G}_n$  has edge  $(i, j)$ , where  $i < j$ , then we set  $a_i <_{\mathcal{A}_n} d_{i,j}$  and  $a_j <_{\mathcal{A}_n} d_{i,j}$ .

First, we show that the poset  $\mathcal{A}_n$  is a complete lattice. It is sufficient to prove that every non-empty set  $B \subseteq \mathcal{A}_n \setminus \{\perp, \top\}$  possesses supremum inside  $\mathcal{A}_n$ . The set  $B$  satisfies one of the following two cases:

*Case 1.* Assume that  $B \cap D_n \neq \emptyset$ . Choose an element  $d_{i,j} \in B \cap D_n$ . If  $d_{i,j}$  is an upper bound of the set  $B$ , then it is clear that  $\text{sup } B = d_{i,j}$ . Otherwise,  $B$  contains an element  $b$ , which is incomparable with  $d_{i,j}$ . The only element, which is an upper bound for both  $d_{i,j}$  and  $b$ , is the top  $\top$ . Hence,  $\text{sup } B = \top$ .

*Case 2.* Assume that  $B \subseteq A$ . If  $\text{card}(B) \geq 3$ , then choose pairwise different  $a_i, a_j, a_k$  from  $B$ . It is not hard to see that  $\top$  is the only element, which bounds all three elements  $a_i, a_j, a_k$ . Thus,  $\text{sup } B = \top$ . Now suppose that  $B = \{a_i, a_j\}$ , where  $i < j$ . If  $\mathcal{G}_n$  has edge  $(i, j)$ , then  $\text{sup } B = d_{i,j}$ ; otherwise,  $\text{sup } B = \top$ . The remaining subcase  $\text{card}(B) = 1$  is trivial.

Second, we observe the following: if  $\psi$  is an automorphism of the poset  $\mathcal{A}_n$ , then  $\psi \upharpoonright A$  maps  $A$  onto  $A$ . This property is ensured by the following fact. Since

the vertex 0 is connected to every other vertex in the graph  $\mathcal{G}_n$ , the set  $A$  is definable inside  $\mathcal{A}_n$  by the formula

$$\xi(x) = \exists y \exists z [(x \text{ and } y \text{ are incomparable}) \& x < z \& y < z \& z < \top].$$

By employing this observation, in a way similar to Sect. 3.3 of [16], one can prove that for all  $n$  and  $m$ , we have  $\mathcal{A}_n \cong \mathcal{A}_m$  if and only if  $\mathcal{G}_n \cong \mathcal{G}_m$ . Therefore, we deduce:

- If  $n \in X$ , then  $\mathcal{A}_n \cong \mathcal{A}_0$ , and the lattice  $\mathbf{FC}_2(\mathcal{S}_2[\mathcal{A}_n])$  is isomorphic to  $\mathbf{FC}_2(\mathcal{S}_2[\mathcal{A}_0]) \cong \mathcal{A}_0$ . Recall that these lattices are induced by Eq. (3).
- If  $n \notin X$ , then  $\mathcal{A}_n \not\cong \mathcal{A}_0$  and  $\mathbf{FC}_2(\mathcal{S}_2[\mathcal{A}_n]) \not\cong \mathcal{A}_0$ .

After that, a standard technical argument establishes that the set  $\mathbf{Iso}_2$  is  $\Sigma_1^1$ -hard. Theorem 4.2 is proved.  $\square$

#### 4.1 Uncountable Lattices

We give two examples of computable families  $\mathcal{S}$ , which induce uncountable lattices  $\mathbf{FC}_2(\mathcal{S})$ . Our first example is the family  $\mathcal{S}_{\text{cofin}}$ , which contains all cofinite subsets of  $\omega$ . It is well-known that  $\mathcal{S}_{\text{cofin}}$  admits a computable numbering. We look at the lattice  $\mathcal{L} := \mathbf{FC}_2(\mathcal{S}_{\text{cofin}})$ .

Consider a proper subset  $A \subset \omega$ . For every  $x \notin A$ , we have  $A \subseteq \omega \setminus \{x\} \in \mathcal{S}_{\text{cofin}}$ . Thus,

$$A = \bigcap \{d \in \mathcal{S}_{\text{cofin}} : A \subseteq d\}.$$

Hence, the proof of Theorem 4.1 implies that *every* set  $A \subseteq \omega$  is  $\beta\alpha$ -closed. Therefore, we deduce that  $\mathcal{L}$  is isomorphic to the Boolean lattice  $\mathcal{P}(\omega)$ , i.e. the algebra of all subsets of  $\omega$ .

Our second example produces a computable family  $\mathcal{S}_{\text{unit}}$  such that the lattice  $\mathcal{M} := \mathbf{FC}_2(\mathcal{S}_{\text{unit}})$  is isomorphic to the real unit interval  $[0; 1]$ .

Fix an algorithmic bijection  $\psi: [0; 1] \cap \mathbb{Q} \rightarrow \omega$ . We define a computable numbering  $\nu$  as follows. For  $n \in \omega$ ,

$$\nu(n) = \{\psi(q) : \psi^{-1}(n) \leq_{\mathbb{Q}} q \leq_{\mathbb{Q}} 1\}.$$

Let  $\mathcal{S}_{\text{unit}}$  be the computable family indexed by  $\nu$ .

Intuitively speaking, any  $\beta\alpha$ -closed subset  $A \subseteq \omega$  encodes a Dedekind cut. More formally, let  $A$  be a non-empty  $\beta\alpha$ -closed set. Then

$$A = \bigcap \{d \in \mathcal{S}_{\text{unit}} : A \subseteq d\}.$$

Consider the set of rationals

$$X[A] := \{q : q = \inf \psi^{-1}(d) \text{ for some } d \in \mathcal{S}_{\text{unit}} \text{ such that } A \subseteq d\}.$$

Then one can show that for closed sets  $A$  and  $B$ , we have  $A \subseteq B$  if and only if  $\sup X[B] \leq_{\mathbb{R}} \sup X[A]$ . Moreover, one can show that the map  $F: A \mapsto \sup X[A]$  is an anti-isomorphism from the poset of  $\beta\alpha$ -closed sets onto the unit interval  $[0; 1]$ . Therefore, we deduce that the lattice  $\mathcal{M}$  is isomorphic to  $[0; 1]$ .

In conclusion, we formulate the following open question:

**Problem 4.1.** Describe uncountable complete lattices that can be realized as  $\mathbf{FC}_2(\mathcal{S})$  for some family  $\mathcal{S}$ .

We introduce one more example, which is related to Problem 4.1. It is not hard to see that the proof of Theorem 4.1 shows that *any* lattice  $\mathbf{FC}_2(\mathcal{S})$  can be isomorphically embedded (as a poset) into the Boolean lattice  $\mathcal{P}(\omega)$ .

Kunen (see p. 85 of [3] and Corollary 42F of [11]) proved that  $\text{MA} + \neg\text{CH}$  (i.e. Martin’s Axiom, together with the negation of the continuum hypothesis) imply that any uncountable subset of  $\mathcal{P}(\omega)$  with no uncountable antichains must contain both ascending and descending infinite sequences with respect to  $\subseteq$ .

Therefore,  $\text{MA} + \neg\text{CH}$  imply that every uncountable successor ordinal  $\alpha < (2^{\aleph_0})^+$  is a complete lattice, which is not isomorphic to any  $\mathbf{FC}_2(\mathcal{S})$ .

## 5 Further Discussion

First, we note that Theorem 4.2 leaves the following question open:

**Problem 5.1.** Find the complexity of the index set  $\mathbf{Iso}_2$ . More formally, what is the many-one degree of  $\mathbf{Iso}_2$ ?

Recall that a computable family  $\mathcal{S}$  can induce the lattice  $\mathbf{FC}_2(\mathcal{S})$  of cardinality continuum. Thus, intuitively speaking, in order to work towards a solution of Problem 5.1, one needs to develop a better understanding of Problem 4.1. In particular, the following question seems to be interesting on its own:

**Problem 5.2.** Find an uncountable complete lattice  $\mathcal{L}$  such that ZFC proves that  $\mathcal{L}$  is not isomorphic to any  $\mathbf{FC}_2(\mathcal{S})$ . In particular, does ZFC prove that the ordinal  $\omega_1 + 1$  is not isomorphic to any  $\mathbf{FC}_2(\mathcal{S})$ ?

## A Proof of Theorem 3.3

First, we need to show that the set  $\mathbf{Iso}_1$  is  $\Pi_3^0$ . By Theorem 3.1, the lattices  $\mathbf{FC}_1(\mathcal{T}_i)$  and  $\mathbf{FC}_1(\mathcal{T}_j)$  are isomorphic if and only if the cardinalities of  $\mathcal{T}_i$  and  $\mathcal{T}_j$  are the same. Semi-formally, this condition can be described as follows.

For every natural number  $n \geq 2$ , the following two conditions are equivalent:

- The family  $\mathcal{T}_i$  contains at least  $n$  different sets. This can be re-written as follows: There exist indices  $k_1, k_2, \dots, k_n$  such that the sets  $\theta_i(k_m) = W_{h_0(i, k_m)}$ ,  $1 \leq m \leq n$ , are pairwise different.
- The family  $\mathcal{T}_j$  contains at least  $n$  different sets.

Since the set  $\{(k, \ell) : W_k \neq W_\ell\}$  is  $\Sigma_2^0$  (see, e.g., Corollary 4.1.8 in [21]), a standard argument shows that our set  $\mathbf{Iso}_1$  is  $\Pi_3^0$ .

Second, suppose that  $X \subseteq \omega$  is an arbitrary  $\Pi_3^0$  set. We fix an index  $i_0$  such that the family  $\mathcal{T}_{i_0}$  is infinite. In order to show that the set  $\mathbf{Iso}_1$  is  $\Pi_3^0$ -complete, it is sufficient to build a uniform sequence of computable numberings  $\{\nu_n\}_{n \in \omega}$  with the following property: for every  $n \in \omega$ ,

(†)  $n \in X$  if and only if the family  $\mathcal{S}_n := \{\nu_n(k) : k \in \omega\}$  is infinite.

Note that Theorem 3.1 and the property (†) together ensure that  $n \in X$  if and only if  $\mathbf{FC}_1(\mathcal{S}_n) \cong \mathbf{FC}_1(\mathcal{T}_{i_0})$ . After building  $\nu_n$ , the rest of the proof is just a standard technical argument.

Choose a computable ternary relation  $R(n, x, y)$  such that

$$n \notin X \Leftrightarrow \exists x \exists^\infty y R(n, x, y). \tag{4}$$

Such a relation  $R$  can be recovered from, e.g., Theorem 4.3.11 in [21].

Fix a natural number  $n$ . For an index  $k \in \omega$ , the construction of the c.e. set  $\nu_n(k)$  proceeds in stages.

At stage 0, we define  $\nu_n(k)[0] := \{\langle k, \ell \rangle : \ell \in \omega\}$ .

At a stage  $s + 1$ , for each  $k \leq s$ , we proceed as follows. If the predicate  $R(n, k, s)$  is true, then we enumerate all numbers  $m \leq \langle k, s \rangle$  into every set  $\nu_n(\ell)$  for  $\ell > k$ .

The construction is described. It is not hard to show that the obtained numberings  $\nu_n$  are uniformly computable.

We verify that the numbering  $\nu_n$  satisfies (†). First, assume that the number  $n$  does not belong to  $X$ . By Eq. (4), we choose the least index  $k_0$  such that  $\exists^\infty s R(n, k_0, s)$ . Then at every stage  $s + 1 > k_0$ , if the condition  $R(n, k_0, s)$  holds, then each set  $\nu_n(\ell)$ , where  $\ell > k_0$ , obtains more new elements. This implies that for every  $\ell > k_0$ , we have  $\nu_n(\ell) = \omega$ . Thus, the family  $\mathcal{S}_n$  contains at most  $k_0 + 2$  elements.

Now assume that  $n \in X$ . Let  $k$  be an arbitrary number. By Eq. (4), one can choose a stage  $s_0$  such that

$$(\forall i \leq k)(\forall s \geq s_0) \neg R(n, i, s).$$

This implies the following: if  $i < k$ , then the element  $\langle k, s_0 \rangle$  belongs to  $A_k \setminus A_i$ ; hence,  $A_k \neq A_i$ . We deduce that the family  $\mathcal{S}_n$  is infinite.

Therefore, the numberings  $\nu_n$  satisfy (†). Theorem 3.3 is proved.  $\square$

## B Proof of Theorem 4.1

(a) Let  $\nu$  be an arbitrary numbering of the family  $\mathcal{S}$ . For a subset  $A \subseteq \omega$ , we have:

$$\begin{aligned} \alpha(A) &= \{n : (\forall x \in A)[(x, n) \in Q_\nu]\} = \{n : A \subseteq \nu(n)\}; \\ \beta \circ \alpha(A) &= \{x : \forall n[A \subseteq \nu(n) \rightarrow x \in \nu(n)]\} = \bigcap \{\nu(n) : A \subseteq \nu(n)\} = \\ &= \bigcap \{d \in \mathcal{S} : A \subseteq d\}. \end{aligned}$$

Therefore, the  $\beta\alpha$ -closure of  $A$  does not depend on the choice of  $\nu$ . Hence, it is easy to show that the isomorphism type of  $\mathbf{FC}_2(\mathcal{S})$  also does not depend on  $\nu$ .

(b) Let  $\mathcal{L}$  be a countable complete lattice. By the Basic Theorem on Concept Lattices, for the formal context  $\mathbb{K} = (\mathcal{L}, \mathcal{L}, \leq_{\mathcal{L}})$ , the lattice  $\mathbb{L}(\mathbb{K})$  is isomorphic to  $\mathcal{L}$ .

Fix a bijection  $\psi$  from  $\mathcal{L}$  onto  $\omega$ . We consider a countable family

$$\mathcal{S}_{\mathcal{L}} = \{\{k \in \omega : \psi^{-1}(k) \leq_{\mathcal{L}} a\} : a \in \mathcal{L}\}.$$

Using Eq. (1), it is not hard to show that the partial order  $(\mathcal{S}_{\mathcal{L}}, \subseteq)$  is isomorphic to the lattice  $\mathbb{L}(\mathbb{K})$ .

On the other hand, for an arbitrary set  $A \subseteq \omega$ , we have:

$$\bigcap \{d \in \mathcal{S}_{\mathcal{L}} : A \subseteq d\} = \bigcap \{\{k : \psi^{-1}(k) \leq_{\mathcal{L}} a\} : a \in \mathcal{L}, \\ a \text{ is an upper bound of } \psi^{-1}(A)\} = \{k : \psi^{-1}(k) \leq_{\mathcal{L}} \sup_{\mathcal{L}} \psi^{-1}(A)\} \in \mathcal{S}_{\mathcal{L}}.$$

Hence, the lattice  $\mathbf{FC}_2(\mathcal{S}_{\mathcal{L}})$  is isomorphic to  $(\mathcal{S}_{\mathcal{L}}, \subseteq)$ , which is, in turn, isomorphic to  $\mathcal{L}$ . Theorem 4.1 is proved.  $\square$

## References

1. Ash, C.J., Knight, J.F.: Computable structures and the hyperarithmetical hierarchy, Stud. Logic Found. Math., vol. 144. Elsevier Science B.V., Amsterdam (2000)
2. Badaev, S.A., Goncharov, S.S.: The theory of numberings: open problems. In: Cholak, P., Lempp, S., Lerman, M., Shore, R. (eds.) Computability Theory and Its Applications. Contemporary Mathematics, vol. 257, pp. 23–38. American Mathematical Society, Providence (2000). <https://doi.org/10.1090/conm/257/04025>
3. Baumgartner, J.E.: Chains and antichains in  $\mathcal{P}(\omega)$ . J. Symb. Log. **45**(1), 85–92 (1980). <https://doi.org/10.2307/2273356>
4. Bazhenov, N., Harrison-Trainor, M., Kalimullin, I., Melnikov, A., Ng, K.M.: Automatic and polynomial-time algebraic structures. J. Symb. Log. **84**(4), 1630–1669 (2019). <https://doi.org/10.1017/jsl.2019.26>
5. Calvert, W., Knight, J.F.: Classification from a computable viewpoint. Bull. Symb. Log. **12**(2), 191–218 (2006). <https://doi.org/10.2178/bsl/1146620059>
6. Ershov, Y.L.: Theory of Numberings. Nauka, Moscow (1977). (in Russian)
7. Ershov, Y.L.: Theory of Numberings. In: Griffor, E.R. (ed.) Handbook of Computability Theory. Studies in Logic and the Foundations of Mathematics, vol. 140, pp. 473–503. North-Holland, Amsterdam (1999). [https://doi.org/10.1016/S0049-237X\(99\)80030-5](https://doi.org/10.1016/S0049-237X(99)80030-5)
8. Ershov, Y.L.: Necessary isomorphism conditions for Rogers semilattices of finite partially ordered sets. Algebra Logic **42**(4), 232–236 (2003). <https://doi.org/10.1023/A:1025005309632>
9. Ershov, Y.L.: Rogers semilattices of finite partially ordered sets. Algebra Logic **45**(1), 26–48 (2006). <https://doi.org/10.1007/s10469-006-0004-9>
10. Fokina, E.B., Harizanov, V., Melnikov, A.: Computable model theory. In: Downey, R. (ed.) Turing’s Legacy: Developments from Turing’s Ideas in Logic. Lectures Notes Logic, vol. 42, pp. 124–194. Cambridge University Press, Cambridge (2014). <https://doi.org/10.1017/CBO9781107338579.006>
11. Fremlin, D.H.: Consequences of Martin’s Axiom. Cambridge Tracts in Mathematics, vol. 84. Cambridge University Press, Cambridge (1984). <https://doi.org/10.1017/CBO9780511896972>

12. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, Berlin (1999). <https://doi.org/10.1007/978-3-642-59830-2>
13. Gödel, K.: Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. Monatshefte für Mathematik und Physik **38**(1), 173–198 (1931). <https://doi.org/10.1007/BF01700692>
14. Goncharov, S.S., Knight, J.F.: Computable structure and non-structure theorems. Algebra Logic **41**(6), 351–373 (2002). <https://doi.org/10.1023/A:1021758312697>
15. Goncharov, S.S., Sorbi, A.: Generalized computable numerations and nontrivial Rogers semilattices. Algebra Logic **36**(6), 359–369 (1997). <https://doi.org/10.1007/BF02671553>
16. Hirschfeldt, D.R., Khossainov, B., Shore, R.A., Slinko, A.M.: Degree spectra and computable dimensions in algebraic structures. Ann. Pure Appl. Logic **115**(1–3), 71–113 (2002). [https://doi.org/10.1016/S0168-0072\(01\)00087-2](https://doi.org/10.1016/S0168-0072(01)00087-2)
17. Kleene, S.C.: On notation for ordinal numbers. J. Symb. Log. **3**(4), 150–155 (1938). <https://doi.org/10.2307/2267778>
18. Kolmogorov, A.N., Uspenskii, V.A.: On the definition of an algorithm. Uspehi Mat. Nauk **13**(4), 3–28 (1958). in Russian
19. McLaughlin, T.G.: The family of all recursively enumerable classes of finite sets. Trans. Am. Math. Soc. **155**(1), 127–136 (1971). <https://doi.org/10.2307/1995467>
20. Rogers, H.: Gödel numberings of partial recursive functions. J. Symb. Log. **23**(3), 331–341 (1958). <https://doi.org/10.2307/2964292>
21. Soare, R.I.: Turing Computability. TAC, Springer, Heidelberg (2016). [https://doi.org/10.1007/978-3-642-31933-4\\_17](https://doi.org/10.1007/978-3-642-31933-4_17)
22. Uspenskii, V.A.: Systems of denumerable sets and their enumeration. Dokl. Akad. Nauk SSSR **105**, 1155–1158 (1958). in Russian
23. Wehner, S.: Computable enumeration and the problem of repetition. Ph.D. thesis, Simon Fraser University (1995)
24. Wille, R.: Restructuring lattice theory: an approach based on hierarchies of concepts. In: Rival, I. (ed.) Ordered Sets, pp. 445–470. D. Reidel Publishing Company, Dordrecht (1982). [https://doi.org/10.1007/978-94-009-7798-3\\_15](https://doi.org/10.1007/978-94-009-7798-3_15)





# Computing Connected- $k$ -Subgraph Cover with Connectivity Requirement

Pengcheng Liu<sup>1</sup>, Zhao Zhang<sup>2</sup>(✉) , Yingli Ran<sup>2</sup>, and Xiaohui Huang<sup>3</sup> 

<sup>1</sup> College of Computer Science and Artificial Intelligence, Wenzhou University, Wenzhou 325035, China

[pengcheng.liu@wzu.edu.cn](mailto:pengcheng.liu@wzu.edu.cn)

<sup>2</sup> College of Mathematics and Computer Science, Zhejiang Normal University, Jinhua 321004, Zhejiang, China

[hxhzz@sina.com](mailto:hxhzz@sina.com), [ranyingli@zjnu.edu.cn](mailto:ranyingli@zjnu.edu.cn)

<sup>3</sup> Library and Information Center, Zhejiang Normal University, Jinhua 321004, Zhejiang, China

**Abstract.** In this paper, we design an  $O(2^{O(\sqrt{t} \log t)} |V|^{O(1)})$  time subexponential FPT algorithm for  $\text{MinCkSC}_{con}$  on an  $H$ -minor free-graph, where  $t$  is an upper bound of solution size.

**Keywords:** Connected- $k$ -subgraph cover · Connected version · Minor-free graph · FPT

## 1 Introduction

This paper studies the minimum connected- $k$ -subgraph cover problem with connectivity requirement ( $\text{MinCkSC}_{con}$ ).

The  $\text{MinCkSC}$  problem was proposed because of a security consideration in a wireless sensor network (WSN) [44]. Given a graph  $G = (V, E)$  and an integer  $k$ , the goal of  $\text{MinCkSC}$  is to find a minimum vertex subset  $C$  such that each connected component of  $G - C$  contains at most  $k - 1$  vertices. The  $\text{MinCkSC}$  problem, under the name of the *minimum  $k$ -vertex separator problem* ( $\text{Min}k\text{VS}$ ), was also studied for the sake of parallel computing [19, 26].

If furthermore, it is required that the connected  $k$ -subgraph cover  $C$  induces a connected subgraph, then the problem is denoted as  $\text{MinCkSC}_{con}$ . Considering connectivity in this problem is because of information sharing issues in a WSN [27]. In this paper, we study FPT algorithm for  $\text{MinCkSC}_{con}$  on  $H$ -minor-free graphs, where  $|V(H)|$  is upper bounded by a constant.

### 1.1 Related Works

Note that there is a problem which is closely related with  $\text{MinCkSC}$ : the *minimum  $k$ -path vertex cover problem* ( $\text{MinVCP}_k$ ). A vertex set  $C$  is called a  $\text{VCP}_k$

---

This research work is supported in part by NSFC (U20A2068, 11771013), and ZJNSFC (LD19A010001).

set if every path on  $k$  vertices has at least one vertex in  $C$ . The  $\text{MinVCP}_k$  problem was first proposed by Novotný [30] also because of a consideration on the security of WSNs. For  $k = 2, 3$ ,  $\text{MinCkSC}$  and  $\text{MinVCP}_k$  coincide. In particular, both  $\text{MinC2SC}$  and  $\text{MinVCP}_2$  are the classic *minimum vertex cover problem* ( $\text{MinVC}$ ) [3, 20], which still receives a lot of attention in recent years [4, 8, 10, 23, 31]. Since  $\text{MinCkSC}$  and  $\text{MinVCP}_k$  and their connected variants are all NP-hard for  $k \geq 2$ , most researches with theoretical guarantees focus on approximation algorithms and FPT algorithms. In the following, we distinguish the weighted version and the cardinality version of these problems by whether using “W” in their abbreviations.

Note that both  $\text{MinWCkSC}$  and  $\text{MinWVCP}_k$  are special cases of the *minimum weight set cover problem* ( $\text{MinWSC}$ ) [29, 44], so they admit  $k$ -approximation. Improvement on the ratio needs deeper exploration of graph structures. Compared with the extensive studies on the  $\text{MinVCP}_k$  problem [7, 8, 21, 32, 39, 40, 43, 45], studies on  $\text{MinCkSC}$  are relatively less.

The terminology  $\text{MinWCkSC}$  was proposed by Zhang *et al.* in [44], and a  $(k-1)$ -approximation algorithm was designed on graphs with girth at least  $k$ . Recently, Liu *et al.* [29] relaxed the girth requirement from  $k$  to  $2k/3$ . For  $\text{MinCkSC}$  (the cardinality version), Lee [26] presented an  $O(\log k)$ -approximation algorithm which runs in time  $2^{O(k^3 \log k)} n^2 \log n + n^{O(1)}$ . For a more general class of problems called  $\mathcal{H}$ -free node-deletion problem which includes both  $\text{MinVCP}_k$  and  $\text{MinCkSC}$ , Li *et al.* [28] present a PTAS on disk graphs. Li *et al.* [27] were the first to study the connected version of the problem, and provided a  $k$ -approximation algorithm for  $\text{MinCkSC}_{con}$ .

Parameterized algorithms for  $\text{MinVCP}_k$  mainly focus on small  $k$ . There are a lot of works on  $\text{MinVCP}_3$  [12, 22, 36, 42] (recall that  $\text{MinVCP}_3$  coincides with  $\text{MinC3SC}$ ). Currently, the best known running time of FPT algorithms for  $\text{MinVC}$  and  $\text{MinVCP}_3$  in terms of solution size  $t$  are  $O^*(1.2738^t)$  [13] and  $O^*(1.713^t)$  [33], respectively. For the connected version with  $k = 2$ , Cygan [14] presented an FPT algorithm for  $\text{MinCVC}$  with running time  $O(2^t n^{O(1)})$ . Bai *et al.* [2] studied parameterized algorithm in terms of treewidth  $\omega$ , presenting a  $3^\omega n^{O(1)}$ -time algorithm for  $\text{MinVCP}_3$  and a  $4^\omega n^{O(1)}$ -time randomized algorithm for  $\text{MinCVCP}_3$ . Parameterized algorithms for  $\text{MinVCP}_k$  with  $k = 4, 5, 6, 7$  are emerging recently [11, 34, 35, 37], but works on general  $k$  are rare. Note that  $\text{MinVCP}_k$  and  $\text{MinCkSC}$  can be considered as special cases of a so-called *graph modification problem* studied by Cai in [9], which has been shown to have an  $O^*(k^t)$  time FPT algorithm. To improve the time complexity, one has to explore graph structures of these problems. As far as we know, there is no study on parameterized algorithm for  $\text{MinCkSC}_{con}$  or  $\text{MinCVCP}_k$  for general  $k$ .

## 1.2 Contribution

In this paper, we study  $\text{MinCkSC}_{con}$  for general constant  $k$ . The contributions are summarized as follows.

We present an  $O(\omega(2(k-1)\omega)^{3\omega})|V|$ -time algorithm for  $\text{MinCkSC}_{con}$ , where  $\omega$  is the treewidth of the graph. The algorithm is a dynamic programming, based

on a nice tree-decomposition. Note that connectivity is a global feature. The challenge of designing the algorithm lies in how to extend partial solutions step by step, keeping enough information to guarantee global connectivity, while at the same time limiting the table size of the dynamic programming. Based on the above result, we prove that  $\text{MinCkSC}_{con}$  has an  $O(\sqrt{t}^{O(\sqrt{t})}|V|)$ -time algorithm on  $H$ -minor-free graphs, where  $t$  is an upper bound for the solution size.

## 2 Algorithm for $\text{MinCkSC}_{con}$ in Terms of Treewidth

### 2.1 Basic Notations

Firstly, we introduce the definition of *tree decomposition*.

**Definition 1 (Tree Decomposition).** For a given graph  $G = (V, E)$ , a pair  $(\{X_i : i \in I\}, T = (I, F))$  is a *tree decomposition*, if the following conditions hold:

- (i)  $\bigcup_{i \in I} X_i = V$ ,
- (ii) every edge  $e \in E$  belongs to some  $G[X_i]$ ,
- (iii)  $T$  is a tree on vertex set  $I$  and edge set  $F$ , such that  $X_i \cap X_k \subseteq X_j$  for any  $j$  lying on the path from  $i$  to  $k$  in  $T$ .

Each  $X_i$  is called a *bag*. The *treewidth* of  $G$  is defined as

$$\omega(G) = \min_{\substack{\text{all tree decomposition} \\ (\{X_i : i \in I\}, T = (I, F))}} \max_{i \in I} (|X_i| - 1).$$

Our algorithm needs *nice tree decomposition* defined as follows. For clarity, we use *nodes* to refer to vertices of  $T$  and use *vertices* to refer to vertices of  $G$ .

**Definition 2 (Nice tree decomposition).** For a graph  $G = (V, E)$ , a tree decomposition is *nice* if nodes in  $I$  can be partitioned into four types as follows:

- (i) *leaf node*: a node in  $T$  without children;
- (ii) *introduce node*: a node  $i$  which has exactly one child  $j$  such that  $X_j \subseteq X_i$  and  $X_i \setminus X_j = \{v\}$  for some vertex  $v \in V$ ;
- (iii) *forget node*: a node  $i$  which has exactly one child  $j$  such that  $X_j = X_i \cup \{v\}$  for some vertex  $v \in V$ ;
- (iv) *join node*: a node  $i$  which has exactly two children  $j, l$  such that  $X_i = X_j = X_l$ .

Bodlaender proved in [6] that for any graph with treewidth  $\omega$ , a tree decomposition can be computed in time  $O(2^{\Theta(\omega^3)}|V|)$ , and given a tree decomposition, one can transform it into a nice tree decomposition in time  $O(|V| + |E|)$ . Note that each non-leaf bag is a separator of graph  $G$  [16].

## 2.2 MinCkSC<sub>con</sub> on Bounded Treewidth Graphs

In this section, we present a dynamic programming for the computation of MinCkSC<sub>con</sub>.

For a nice tree decomposition  $(\{X_i : i \in I\}, T)$ , a subtree of  $T$  rooted at  $i$  which contains all descendants of  $i$  is denoted as  $T_i$ . The subgraph of  $G$  induced by all vertices of those bags corresponding to those nodes of  $T_i$  is denoted as  $G_i = G[\bigcup_{j \in V(T_i)} X_j]$ .

**Status Variables:** For each node  $i \in I$ , the dynamic programming will record a set of optimal values with respect to all possible tokens of the form  $(F, P_F, P_{X_i \setminus F})$ , where  $F \subseteq X_i$ ,  $P_F$  is a partition of  $F$ ,  $P_{X_i \setminus F}$  consists of a partition of  $X_i \setminus F$ , and each part of  $P_{X_i \setminus F}$  is associated with an integer smaller than  $k$ . Define status variable  $c(i, F, P_F, P_{X_i \setminus F})$  to be the cardinality of a minimum vertex set  $C \subseteq V(G_i)$  which is compatible with token  $(F, P_F, P_{X_i \setminus F})$ , where *compatible* means

- (i)  $C \cap X_i = F$ ;
- (ii)  $P_F$  is the partition of  $F$  corresponding to those connected components of  $G_i[C]$ ;
- (iii)  $P_{X_i \setminus F}$  consists of the partition of  $X_i \setminus F$  corresponding to those connected components of  $G_i - C$  which has non-empty intersection with  $X_i$ , and for each part of the partition, the integer associated with this part is the cardinality of the connected component of  $G_i - C$  containing this part.

We shall use  $P_F(v)$  (resp.  $P_{X_i \setminus F}(v)$ ) to denote the part of  $P_F$  (resp.  $P_{X_i \setminus F}$ ) which contains vertex  $v$ , and use  $k_D$  to denote the integer associated with a part  $D$  of the partition of  $P_{X_i \setminus F}$ .

**Initial Condition:** The dynamic programming starts computing from leaf nodes. For a leaf node  $i \in I$ , if  $F$  is compatible with token set  $(F, P_F, P_{X_i \setminus F})$ , then  $c(i, F, P_F, P_{X_i \setminus F}) = |F|$ , otherwise  $c(i, F, P_F, P_{X_i \setminus F}) = \infty$ .

**Transition Formula:** We distinguish the transition formula into three cases depending on the type of the internal node.

**Introduce Node:** Suppose  $i$  is an introduce node. Let  $j$  be the only child of  $i$  and denote by  $v$  the unique vertex in  $X_i \setminus X_j$ . A triple  $(F, P_F, P_{X_i \setminus F})$  is said to be a *valid token* for node  $i$  if

- (a<sup>(in)</sup>) every part of  $P_{X_i \setminus F}$  has cardinality at most  $k - 1$ ,
- and exactly one of the following two conditions is satisfied:

- (b<sub>1</sub><sup>(in)</sup>) if  $v \in F$ , then  $N_G(v) \cap F \subseteq P_F(v)$ ;
- (b<sub>2</sub><sup>(in)</sup>) if  $v \notin F$ , then  $N_G(v) \cap (X_i \setminus F) \subseteq P_{X_i \setminus F}(v)$ .

Superscript (in) indicates that the condition is for an introduce node. Condition (a<sup>(in)</sup>) is necessary for  $C$  to be a feasible CkSC<sub>con</sub>. The reason for condition (b<sub>1</sub><sup>(in)</sup>) is because if  $v \in F$ , then any neighbor of  $v$  in  $F$  must belong to the connected component of  $G_i[C]$  containing  $v$ . Similar reason for condition (b<sub>2</sub><sup>(in)</sup>).

Note that in the case of introduce node,  $G_j = G_i - v$  due to condition (iii) in the definition of tree decomposition (Definition 1). For a valid token

$(F, P_F, P_{X_i \setminus F})$ , we say that token  $(F', P_{F'}, P_{X_j \setminus F'})$  for node  $j$  is *compatible* with  $(F, P_F, P_{X_i \setminus F})$  if one of the following two conditions are met:

$(\mathbf{c}_1^{(in)})$  If  $v \in F$ , then  $F' = F \setminus \{v\}$ ,  $P_F = (P_{F'} \setminus \{P_{F'}(u) : u \in F' \cap N_G(v)\}) \cup \{D\}$  where  $D = \left( \bigcup_{u \in F' \cap N_G(v)} P_{F'}(u) \right) \cup \{v\}$ , and  $P_{X_j \setminus F'} = P_{X_i \setminus F}$ .

$(\mathbf{c}_2^{(in)})$  If  $v \notin F$ , then  $F' = F$ ,  $P_F = P_{F'}$ , the partition of  $P_{X_i \setminus F}$  is  $(P_{X_j \setminus F'} \setminus \{P_{X_j \setminus F'}(u) : u \in (X_j \setminus F') \cap N_G(v)\}) \cup \{D'\}$ , where  $D' = \left( \bigcup_{u \in (X_j \setminus F') \cap N_G(v)} P_{X_j \setminus F'}(u) \right) \cup \{v\}$ , and  $k_{D'} = 1 + \sum_{D \in \{P_{X_j \setminus F'}(u) : u \in (X_j \setminus F') \cap N_G(v)\}} k_D$  must satisfy  $k_{D'} \leq k - 1$ .

The transition formula for a valid token  $(F, P_F, P_{X_i \setminus F})$  of an introduce node  $i$  is

$$c(i, F, P_F, P_{X_i \setminus F}) = \min_{\substack{(F', P_{F'}, P_{X_j \setminus F'}) : \text{token for node } j \\ \text{compatible with } (F, P_F, P_{X_i \setminus F})}} c(j, F', P_{F'}, P_{X_j \setminus F'}) + 1_{v \in F},$$

where  $1_{v \in F} = 1$  if  $v \in F$  and  $1_{v \in F} = 0$  if  $v \notin F$ .

**Forget Node:** Suppose  $i$  is a forget node,  $j$  is the only child of  $i$ , and  $v$  is the only vertex in  $X_j \setminus X_i$ . A triple  $(F, P_F, P_{X_i \setminus F})$  is said to be a *valid token* for node  $i$  if

$(\mathbf{a}^{(for)})$  each part of  $P_{X_i \setminus F}$  has at most  $k - 1$  vertices, and one of the following two conditions is satisfied:

$(\mathbf{b}_1^{(for)})$  all vertices in  $N_G(v) \cap F$  belong to a same part of  $P_F$ , or

$(\mathbf{b}_2^{(for)})$  all vertices in  $N_G(v) \cap (X_i \setminus F)$  belong to a same part of  $P_{X_i \setminus F}$ .

Superscript (for) indicates that the condition is for a forget node. Note that different from the case for introduce node, in which exactly one condition of  $(b_1^{(in)})$  and  $(b_2^{(in)})$  can occur, for the case of forget node, conditions  $(b_1^{(for)})$  and  $(b_2^{(for)})$  can both occur.

For the case of forget node,  $G_i = G_j$ . For a valid token  $(F, P_F, P_{X_i \setminus F})$ , we say that a token  $(F', P_{F'}, P_{X_j \setminus F'})$  for node  $j$  is *compatible* with  $(F, P_F, P_{X_i \setminus F})$  if one of the following two conditions are met:

$(\mathbf{c}_1^{(for)})$  All vertices of  $N_G(v) \cap F$  belong to a same part of  $P_F$ ,  $F' = F \cup \{v\}$ ,  $P_F = (P_{F'} \setminus \{P_{F'}(v)\}) \cup \{P_{F'}(v) \setminus \{v\}\}$ , and  $P_{X_i \setminus F} = P_{X_j \setminus F'}$ .

$(\mathbf{c}_2^{(for)})$  All vertices of  $N_G(v) \cap (X_i \setminus F)$  belong to a same part of  $P_{X_i \setminus F}$ ,  $F' = F$ ,  $P_{F'} = P_F$ ,  $P_{X_i \setminus F} = (P_{X_j \setminus F'} \setminus \{P_{X_j \setminus F'}(v)\}) \cup \{P_{X_j \setminus F'}(v) \setminus \{v\}\}$ , and  $k_{P_{X_j \setminus F'}(v)} \leq k - 1$ .

The transition formula for a valid token  $(F, P_F, P_{X_i \setminus F})$  of a forget node  $i$  is

$$c(i, F, P_F, P_{X_i \setminus F}) = \min_{\substack{(F', P_{F'}, P_{X_j \setminus F'}) : \text{token of node } j \\ \text{compatible with } (F, P_F, P_{X_i \setminus F})}} c(j, F', P_{F'}, P_{X_j \setminus F'}).$$

**Join Node:** Suppose  $i \in I$  is a join node with two children  $j$  and  $l$ . A triple  $(F, P_F, P_{X_i \setminus F})$  is a valid token if each part of  $P_{X_i \setminus F}$  has at most  $k - 1$  vertices.

In this case,  $X_i = X_j = X_l$  and  $G_i = G_j \cup G_l$ . Furthermore, by condition (iii) of tree decomposition, it can be observed that

$$V(G_j) \cap V(G_l) \text{ is exactly } X_i. \tag{1}$$

For a valid token  $(F, P_F, P_{X_i \setminus F})$  of node  $i$ , we say that a token  $(F', P_{F'}, P_{X_i \setminus F'})$  of node  $j$  and a token  $(F'', P_{F''}, P_{X_i \setminus F''})$  of node  $l$  are compatible with  $(F, P_F, P_{X_i \setminus F})$  if the following conditions are satisfied:

(c<sup>(join)</sup>)  $F = F' = F''$ ;

(d<sup>(join)</sup>) Every part of  $P_{F'}$  (as well as every part of  $P_{F''}$ ) is completely contained in some part of  $P_F$ . Furthermore, if there are a part  $D'$  of  $P_{F'}$  and a part  $D''$  of  $P_{F''}$  which have non-empty intersection, then  $D'$  and  $D''$  belong to a same part of  $P_F$ ;

(e<sup>(join)</sup>) Every part of  $P_{X_j \setminus F'}$  (as well as every part of  $P_{X_l \setminus F''}$ ) is completely contained in some part of  $P_{X_i \setminus F}$ . If there is a part  $D'$  of  $P_{X_j \setminus F'}$  and a part  $D''$  of  $P_{X_l \setminus F''}$  with  $D' \cap D'' \neq \emptyset$ , then  $D = D' \cup D''$  is a part of  $P_{X_i \setminus F}$ , and furthermore, the integer associated with  $D$  satisfies  $k_D = k_{D'} + k_{D''} - |D' \cap D''| \leq k - 1$ .

The transition formula for a valid token  $(F, P_F, P_{X_i \setminus F})$  of a join node  $i$  is

$$c(i, F, P_F, P_{X_i \setminus F}) = \min_{\substack{(F', P_{F'}, P_{X_j \setminus F'}), \\ (F'', P_{F''}, P_{X_l \setminus F''}): \\ \text{tokens compatible with} \\ (F, P_F, P_{X_i \setminus F})}} \{c(j, F', P_{F'}, P_{X_j \setminus F'}) + c(l, F'', P_{F''}, P_{X_l \setminus F''}) - |F|\}.$$

In the above, we only give out transition formula for valid tokens. If  $(F, P_F, P_{X_i \setminus F})$  is an invalid token for node  $i$ , or there are no tokens for his child (or children) which are compatible with  $(F, P_F, P_{X_i \setminus F})$ , then set

$$c(i, F, P_F, P_{X_i \setminus F}) = \infty.$$

**Theorem 1.** *Given a graph  $G = (V, E)$  with tree-width  $\omega$ , starting from leaf nodes of a nice tree decomposition of  $G$ , recursively use the above transition formulas in a bottom-up manner, when the dynamic programming arrives at the root node  $r$ , the minimum vertex set over all possible tokens  $(F, P_F, P_{X_r \setminus F})$  for  $r$  is an optimal solution for the  $\text{MinCkSC}_{con}$  instance. The time complexity is  $O((2^{\Theta(\omega^3)} + \omega(2(k-1)\omega)^{3\omega})|V|)$ .*

The detailed proof is put in the full version of this paper. As a corollary of Theorem 1, we have the following result.

**Corollary 1.** *The  $\text{MinCkSC}_{con}$  problem is polynomial time solvable on a graph whose tree-width is bounded above by a constant.*

### 3 Sub-exponential FPT Algorithm for $\text{MinCkSC}_{\text{con}}$ on $H$ -minor-free Graphs

In this section, we present an FPT algorithm for the following parameterized version of the  $\text{MinCkSC}_{\text{con}}$  problem on an  $H$ -free graph  $G$ : For a fixed graph  $H$ , given an  $H$ -free graph  $G$  and an integer  $t$ , is there a  $\text{CkSC}_{\text{con}}$  in  $G$  with size at most  $t$ ? If the answer is yes, then the algorithm is able to find out an optimal solution in time  $f(t)p(n)$ , where  $p(n)$  is a polynomial on the input size  $n$ , and  $f(t)$  is a function depending on  $t$  and  $H$  (but is independent of  $n$ ).

The algorithm makes use of the following result which was due to Demaine and Hajiaghayi [15].

**Lemma 1** ([15]). *Any  $H$ -minor-free graph with tree-width at least  $\omega$  has an  $c(H)\omega \times c(H)\omega$  grid as a minor, where  $c(H)$  is a parameter depending on  $H$ .*

The next lemma gives a lower bound for the size of a  $\text{CkSC}$  in a grid.

**Lemma 2.** *For a moderately large  $g$  (compared with  $k$ ), any  $\text{CkSC}$  on a  $g \times g$  grid has size larger than  $g^2/(2k)$ .*

*Proof.* Suppose  $C$  is a  $\text{CkSC}$  of a  $g \times g$  grid. Note that each  $\lceil \sqrt{k} \rceil \times \lceil \sqrt{k} \rceil$  grid must contain a vertex of  $C$ . A  $g \times g$  grid has at least  $\lfloor \frac{g}{\lceil \sqrt{k} \rceil} \rfloor \times \lfloor \frac{g}{\lceil \sqrt{k} \rceil} \rfloor$  disjoint  $\lceil k \rceil \times \lceil k \rceil$  grids. When  $g$  is moderately large,  $\lfloor \frac{g}{\lceil \sqrt{k} \rceil} \rfloor \times \lfloor \frac{g}{\lceil \sqrt{k} \rceil} \rfloor > \frac{g^2}{2k}$ . Then the lemma follows.

The FPT algorithm is embedded in the proof of the following theorem.

**Theorem 2.** *Suppose  $H$  is a graph with a constant number of vertices and  $k$  is a constant integer. The parameterized version of the  $\text{MinCkSC}_{\text{con}}$  problem on an  $H$ -minor-free graph can be solved in time  $O(\sqrt{t}^{O(\sqrt{t})}|V|)$ .*

*Proof.* As a consequence of Lemma 2, if the minimum  $\text{CkSC}_{\text{con}}$  of graph  $G$  has at most  $g^2/2k$  vertices, then  $G$  cannot have  $g \times g$  grid as a minor. So, for the parameterized version of the  $\text{CkSC}_{\text{con}}$  problem with parameter  $t$ , if the answer is yes, then  $G$  is  $(\sqrt{2kt} \times \sqrt{2kt})$ -grid minor free. By Lemma 1, the tree-width of  $G$  is smaller than  $\sqrt{2kt}/c(H)$ . Combining this with Theorem 1, we could find an optimal solution in time  $O\left(\left(2^{\Theta((\sqrt{2kt}/c(H))^3)} + \frac{\sqrt{2kt}}{c(H)}\left(2(k-1)\frac{\sqrt{2kt}}{c(H)}\right)^{3\frac{\sqrt{2kt}}{c(H)}}\right)|V|\right)$ .

On the other hand, if the computed treewidth is larger than  $\sqrt{2kt}/c(H)$ , then the answer to the instance must be no.

In particular, if  $|V(H)|$  has a constant upper bound, then  $c(H)$  is a constant. Combining this with our assumption that  $k$  is a constant, Theorem 2 follows.

## 4 Conclusion

In this paper, we presented a dynamic programming for  $\text{MinCkSC}_{con}$  the running time of which depends on the treewidth; and obtained an FPT algorithm for  $\text{MinCkSC}_{con}$  on  $H$ -minor-free graphs. Our results hold for general constant  $k$ . There are still some further questions about possibility of extending the work. Can the time be improved via some other design techniques which have been successfully used on the connected vertex cover problem? Is it possible to achieve better results on some special minor free-graphs such as planar graph and outerplanar graphs?

**Acknowledgment.** This research work is supported in part by NSFC (U20A2068, 11771013), and ZJNSFC (LD19A010001).

## References

1. Alon, N., Seymour, P.D., Thomas, R.: A separator theorem for nonplanar graphs. *J. Am. Math. Soc.* **3**(4), 801–808 (1990)
2. Bai, Z., Tu, J., Shi, Y.: An improved algorithm for the vertex cover  $P_3$  problem on graphs of bounded treewidth. *Discrete Math. Theor. Comput. Sci.* **21** (2019)
3. Bar-Yehuda, R., Even, S.: A linear time approximation algorithm for the weighted vertex cover problem. *J. Algorithms* **2**, 198–203 (1981)
4. Bar-Yehuda, R., Censor-Hillel, K., Schwartzman, G.: A distributed  $(2 + \epsilon)$ -approximation for vertex cover in  $O(\log \Delta / \epsilon \log \log \Delta)$  rounds, *J. ACM* **64**(3), 1–11 (2017)
5. Bateni, M., Farhadi, A., Hajiaghayi, M.: Polynomial-time approximation scheme for minimum  $k$ -cut in planar and minor-free graphs, In: Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, pp. 1055–1068 (2019)
6. Bodlaender, H.L.: A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.* **25**(6), 1305–1317 (1996)
7. Brešar, B., Kardoš, F., Katrenič, J., Semanišin, G.: Minimum  $k$ -path vertex cover. *Discret. Appl. Math.* **159**, 1189–1195 (2011)
8. Cabello, S., Gajser, D.: Simple PTAS's for families of graphs excluding a minor. *Discret. Appl. Math.* **189**, 41–48 (2015)
9. Cai, L.: Fixed-parameter tractability of graph modification problems for hereditary properties. *Inf. Process. Lett.* **58**(4), 171–176 (1996)
10. Cai, S., Li, Y., Hou, W., Wang, H.: Towards faster local search for minimum weight vertex cover on massive graphs. *Inf. Sci.* **471**, 64–79 (2019)
11. Červený, R., Suchý, O.: Faster FPT algorithm for 5-path vertex cover. In: Proceedings of 44th International Symposium on Mathematical Foundations of Computer Science (MFCS 2019), vol. 32, pp. 1–13 (2019)
12. Chang, M.S., Chen, L.H., Hung, L.J., Rossmanith, P., Su, P.C.: Fixed-parameter algorithms for vertex cover  $P_3$ . *Discret. Optim.* **19**, 12–22 (2016)
13. Chen, J., Kanj, I.A., Xia, G.: Improved upper bounds for vertex cover. *Theoret. Comput. Sci.* **411**(40–42), 3736–3756 (2010)
14. Cygan, M.: Deterministic parameterized connected vertex cover. In: Fomin, F.V., Kaski, P. (eds.) SWAT 2012. LNCS, vol. 7357, pp. 95–106. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-31155-0\\_9](https://doi.org/10.1007/978-3-642-31155-0_9)



15. Demaine, E.D., Hajiaghayi, M.: Linearity of grid minors in treewidth with applications through bidimensionality. *Combinatorica* **28**(1), 19–36 (2008)
16. Diestel, R.: *Graph Theory*. GTM, vol. 173. Springer, Heidelberg (2017). <https://doi.org/10.1007/978-3-662-53622-3>
17. Fomin, F.V., Lokshtanov, D., Raman, V., Saurabh, S.: Bidimensionality and EPTAS, In: *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011*, pp. 748–759 (2011)
18. Frederickson, G.: Fast algorithms for shortest paths in planar graphs with applications. *SIAM J. Comput.* **16**(6), 1004–1022 (1987)
19. Guruswami, V., Lee, E.: Inapproximability of  $H$ -transversal/packing. *SIAM J. Discret. Math.* **31**(3), 1552–1571 (2017)
20. Hochbaum, D.S.: Approximation algorithms for the set covering and vertex cover problems. *SIAM J. Comput.* **11**(3), 555–556 (1982)
21. Kardoš, F., Katrenič, J., Schiermeyer, I.: On computing the minimum 3-path vertex cover and dissociation number of graphs. *Theoret. Comput. Sci.* **412**, 7009–7017 (2011)
22. Katrenič, J.: A faster FPT algorithm for 3-path vertex cover. *Inf. Process. Lett.* **116**(4), 273–278 (2016)
23. Khot, S., Minzer, D., Safra, M.: Pseudorandom sets in grassmann graph have near-perfect expansion. In: *2018 IEEE 59th Annual Symposium on Foundations of Computer Science, FOCS 2018*, pp. 592–601 (2018)
24. Kloks, T. (ed.): *Treewidth*. LNCS, vol. 842. Springer, Heidelberg (1994). <https://doi.org/10.1007/BFb0045375>
25. Le, H., Zheng, B.: Local search is a PTAS for feedback vertex set in minor-free graphs. *Theoret. Comput. Sci.* **838**, 17–24 (2020)
26. Lee, E.: Partitioning a graph into small pieces with applications to path transversal, In: *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017*, pp. 1546–1558 (2017)
27. Li, X., Zhang, Z., Huang, X.: Approximation algorithms for minimum (weight) connected  $k$ -path vertex cover. *Discret. Appl. Math.* **205**, 101–108 (2016)
28. Li, X., Shi, Y., Huang, X.: PTAS for  $\mathcal{H}$ -free node deletion problems in disk graphs. *Discret. Appl. Math.* **239**, 119–124 (2018)
29. Liu, P., Zhang, Z., Huang, X.: Approximation algorithm for minimum weight connected- $k$ -subgraph cover. *Theoret. Comput. Sci.* **838**, 160–67 (2020)
30. Novotný, M.: Design and analysis of a generalized canvas protocol. In: Samarati, P., Tunstall, M., Posegga, J., Markantonakis, K., Sauveron, D. (eds.) *WISTP 2010*. LNCS, vol. 6033, pp. 106–121. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-12368-9\\_8](https://doi.org/10.1007/978-3-642-12368-9_8)
31. Pourhassan, M., Shi, F., Neumann, F.: Parameterized analysis of multiobjective evolutionary algorithms and the weighted vertex cover problem. *Evol. Comput.* **27**(4), 559–575 (2019)
32. Ran, Y., Zhang, Z., Huang, X., Li, X., Du, D.: Approximation algorithms for minimum weight connected 3-path vertex cover. *Appl. Math. Comput.* **347**, 723–733 (2019)
33. Tsur, D.: Parameterized algorithm for 3-path vertex cover. *Theoret. Comput. Sci.* **783**, 1–8 (2019)
34. Tsur, D.: An  $O^*(2.619^k)$  algorithm for 4-path vertex cover. *Discrete Appl. Math.* **291**, 1–14 (2021)
35. Tsur, D.:  $l$ -path vertex cover is easier than  $l$ -hitting set for small  $l$ , *ArXiv Preprint ArXiv:1906.10523* (2019)

36. Tu, J.: A fixed-parameter algorithm for the vertex cover  $P_3$  problem. *Inf. Process. Lett.* **115**, 96–99 (2015)
37. Tu, J., Jin, Z.: An FPT algorithm for the vertex cover  $P_4$  problem. *Discret. Appl. Math.* **200**, 186–190 (2016)
38. Tu, J., Shi, Y.: An efficient polynomial time approximation scheme for the vertex cover  $P_3$  problem on planar graphs. *Discussiones Math. Graph Theory* **39**, 55–65 (2019)
39. Tu, J., Zhou, W.: A factor 2 approximation algorithm for the vertex cover  $P_3$  problem. *Inf. Process. Lett.* **111**, 683–686 (2011)
40. Tu, J., Zhou, W.: A primal-dual approximation algorithm for the vertex cover  $P_3$  problem. *Theoret. Comput. Sci.* **412**, 7044–7048 (2011)
41. Tu, J., Yang, F.: The vertex cover  $P_3$  problem in cubic graphs. *Inf. Process. Lett.* **113**, 481–485 (2013)
42. Xiao, M., Kou, S.: Exact algorithms for the maximum dissociation set and minimum 3-path vertex cover problems. *Theoret. Comput. Sci.* **657**, 86–97 (2017)
43. Zhang, A., Chen, Y., Chen, Z., Lin, G.H.: Improved approximation algorithms for path vertex covers in regular graphs. [ArXiv: 1811.01162v1](https://arxiv.org/abs/1811.01162v1) (2018)
44. Zhang, Y., Shi, Y.S., Zhang, Z.: Approximation algorithm for the minimum weight connected  $k$ -subgraph cover problem. *Theoret. Comput. Sci.* **535**, 54–58 (2014)
45. Zhang, Z., Li, X., Shi, Y., Nie, H., Zhu, Y.: PTAS for minimum  $k$ -path vertex cover in ball graph. *Inf. Process. Lett.* **119**, 9–13 (2017)



# Analyzing the 3-path Vertex Cover Problem in Planar Bipartite Graphs

Sangram K. Jena and K. Subramani<sup>(✉)</sup>

LDCSEE, West Virginia University, Morgantown, WV, USA  
{sangramkishor.jena,k.subramani}@mail.wvu.edu

**Abstract.** Let  $G = (V, E)$  be a simple graph. A set  $C \subseteq V$  is called a  $k$ -path vertex cover of  $G$ , if each  $k$ -path in  $G$  contains at least one vertex from  $C$ . In the  $k$ -path vertex cover problem, we are given a graph  $G$  and asked to find a  $k$ -path vertex cover of minimum cardinality. For  $k = 3$ , the problem becomes the well-known 3-path vertex cover (3PVC) problem, which has been widely studied, as per the literature. In this paper, we focus on the 3PVC problem in planar bipartite (pipartite) graphs for the most part. We first show that the 3PVC problem is **NP-hard**, even in bipartite graphs in which the degree of all vertices is bounded by 4. We then show that the 3PVC problem on this class of graphs admits a linear time 1.5-approximation algorithm. Finally, we show that the 3PVC problem is **APX-complete** in bipartite graphs. The last result is particularly interesting, since the vertex cover problem in bipartite graphs is solvable in polynomial time.

## 1 Introduction

Given a simple undirected graph  $G = (V, E)$ , the *open neighborhood* (resp. *closed neighborhood*) of a vertex  $v_i \in V$  is defined by  $N(v_i) = \{v_j \in V \mid v_i v_j \in E\}$  (resp.  $N[v_i] = N(v_i) \cup \{v_i\}$ ). The degree of a vertex  $v$  in the graph  $G$  is defined as  $d_G(v) = |N(v)|$ , whereas the maximum degree of a graph is  $\Delta(G) = \max_{v \in V} \{d_G(v)\}$ .

A *vertex cover*  $C$  of  $G$  is a subset of  $V$  such that for each edge  $uv \in E$ , either  $u \in C$  or  $v \in C$ . The (minimum) vertex cover problem asks to find a vertex cover of minimum size in a given graph. One generalization of the vertex cover problem is the  *$k$ -path vertex cover* problem. A  *$k$ -path vertex cover*  $C_k$  of  $G$  is a subset of  $V$  such that each path in  $G$  having  $k$  vertices (path of order  $k$ ) contains at least one vertex from  $C_k$ . In other words,  $C_k$  is called a  $k$ -path vertex cover ( $k$ PVC) of  $G$ , if there does not exist a path of order  $k$  in the induced subgraph  $G' = (V \setminus C_k, E')$ , where an edge  $e \in E$  belongs to  $E'$ , if both its endpoints are in  $V \setminus C_k$ . The (minimum)  $k$ -path vertex cover problem asks to find a vertex subset of minimum size satisfying the  $k$ -path vertex cover property in a given graph  $G$ . For  $k = 3$ , the  $k$ -path vertex cover problem is called the 3-path vertex cover (3PVC) problem.

This research was supported in part by the Air-Force Office of Scientific Research through Grant FA9550-19-1-0177 and in part by the Air-Force Research Laboratory, Rome through Contract FA8750-17-S-7007.

In the 3PVC problem, we are given an undirected, unweighted graph  $G = (V, E)$  and the goal is to find a minimum cardinality set  $V' \subseteq V$ , such that at least one vertex from every two-edge path is in  $V'$ . It is clear that the 3PVC problem is a variant of the well-known vertex cover (VC) problem and a specialization of the  $k$ -path vertex cover problem, discussed in [1]. The 3PVC problem finds applications in several practical domains, including wireless networks and data integrity [1, 6]. Prior work has established the computational difficulty of this problem in general graphs. Indeed, the 3PVC problem is known to be **NP-hard** for planar graphs and bipartite graphs. This paper studies the 3PVC problem in planar bipartite (pipartite) graphs, i.e., the intersection of the above-mentioned graph classes.

The principal contributions of the paper are as follows:

1. A proof that the 3PVC problem is **NP-complete** in pipartite graphs, even with  $\Delta(G) \leq 4$  (Sect. 3).
2. The design and analysis of a linear time 1.5-approximation algorithm for the 3PVC problem in pipartite graphs, with  $\Delta(G) \leq 4$  (Sect. 4).
3. A proof of **APX-completeness** for the 3PVC problem in bipartite graphs (Sect. 5).

The rest of this paper is organized as follows: In Sect. 2, we discuss related work in the literature. The computational complexity of the 3PVC problem in pipartite graphs is detailed in Sect. 3. An approximation algorithm for this problem on a selected class of pipartite graphs is discussed in Sect. 4. In Sect. 5, we show that the 3PVC problem is **APX-complete** in bipartite graphs. Finally, we conclude in Sect. 6 by summarizing our contributions and identifying avenues for future research.

## 2 Related Work

In this section, we discuss the state-of-the-art results of the 3-path vertex cover problem. The generalized version of the 3-path vertex cover (3PVC) problem is the  $k$ -path vertex cover ( $k$ PVC) problem. Motivated by two problems, viz., (i) secure communication in wireless sensor networks [1, 11] and (ii) controlling traffic at street crossings [15], Brešar et al. [1] introduced the  $k$ PVC problem in 2011. For  $k \geq 2$ , Brešar et al. [1] proved that determining  $\psi_k(G)$  (minimum cardinality of a  $k$ PVC) in a graph  $G$  is **NP-hard**. They proved that the problem can be solved in linear time in trees. For  $k = 2$ , the problem is known as the vertex cover (VC) problem in the literature. The VC problem is known to be **NP-hard**, in general [8]. Brešar et al. [1] proved the existence of an  $r$ -approximation algorithm for the VC problem from an  $r$ -approximation algorithm of the  $k$ PVC problem. Note that a  $k$ -approximation algorithm for the  $k$ PVC problem is trivial [1]. The authors also presented several estimations and exact values to provide the upper bound for  $\psi_k(G)$ . They proved  $\psi_3(G) \leq (2 \cdot n + m)/6$  for any graph  $G$  with  $n$  vertices and  $m$  edges. For outerplanar graphs of order  $n$ , they proved  $\psi_3(G) \leq \frac{n}{2}$ . In [13], Tu and Yang proved that the 3PVC problem is **NP-hard**

in cubic planar graphs with girth 3. They also proposed a linear time 1.57-approximation algorithm for the 3PVC problem in cubic graphs. Whether a polynomial-time  $c$ -approximation algorithm exists for the  $k$ PVC problem for  $k \geq 2$  [1, 7] is an open problem.

For the 3PVC problem, many constant factor approximation results are known. Kardoš et al. [7] proposed a polynomial-time randomized approximation algorithm with an expected approximation ratio of  $\frac{23}{11}$ . In [14, 15], Tu and Zhou proposed several approximation algorithms for the weighted  $k$ PVC problem (each vertex has a weight). The two techniques they used were the primal-dual method and graph layering. Zhang et al. [16] considered the  $k$ PVC problem in  $d$ -regular graphs and proposed several approximation results. The 3PVC problem in planar graphs admits an **EPTAS** [12], which means that the problem is not **APX-hard** in bipartite graphs unless **P** = **NP**.

### 3 Computational Complexity

In this section, we reduce the vertex cover (VC) problem in planar graphs to the 3-path vertex cover (3PVC) problem in bipartite graphs via a linear time algorithm. Note that the VC problem in planar graphs with maximum degree three is known to be **NP-hard** [10].

The decision versions of both the problems are defined below.

#### The vertex cover problem in planar graphs (VC-PLA)

Given a planar graph  $G$  having maximum degree three and a positive integer  $k$ , does  $G$  have a VC of size at most  $k$ ?

#### The 3PVC problem in bipartite graphs (3PVC-PB)

Given a bipartite graph  $G$  and a positive integer  $k$ , does there exist a 3PVC of size at most  $k$ ?

**Construction:** The construction from a given instance of a planar graph  $G$  to an instance of a bipartite graph  $G'$  takes place in three steps.

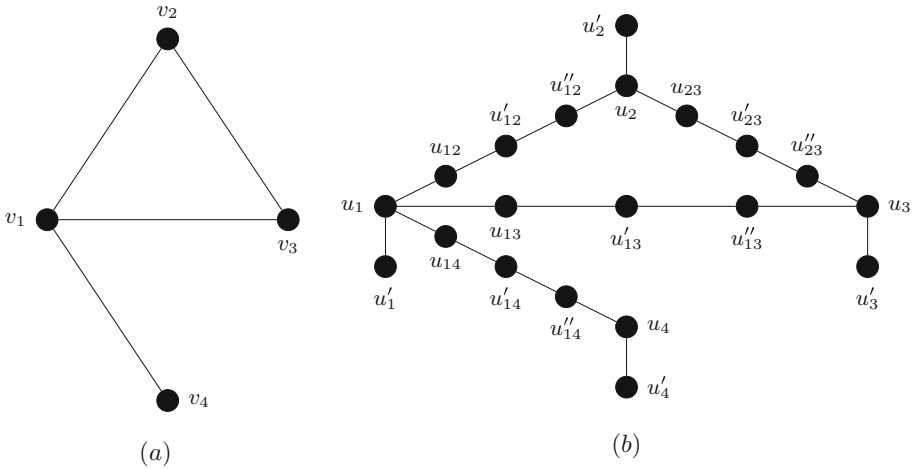
Step 1: For each vertex  $v_i$  in  $G$ , create a corresponding vertex  $u_i$  in  $G'$ .

Step 2: For each vertex  $u_i$  in  $G'$ , create a support vertex  $u'_i$  and put an edge between  $u_i$  and  $u'_i$ .

Step 3: For each edge  $v_i v_j \in E$  in the graph  $G$ , take the corresponding vertices  $u_i$  and  $u_j$  in  $G'$  and put three vertices  $u_{ij}$ ,  $u'_{ij}$ , and  $u''_{ij}$  between them. Now, add four edges in the order  $u_i u_{ij}$ ,  $u_{ij} u'_{ij}$ ,  $u'_{ij} u''_{ij}$ , and  $u''_{ij} u_j$  (see the three vertices and four edges added between  $u_1$  and  $u_2$  in Fig. 1 (b), corresponding to the edge  $v_1 v_2$  of Fig. 1 (a)).

**Lemma 1.** *For a given instance of a planar graph  $G = (V, E)$  ( $\Delta(G) \leq 3$ ), an instance of a bipartite graph  $G' = (V_1, V_2, E')$  ( $\Delta(G') \leq 4$ ) can be constructed in linear time using the above construction.*

*Proof.* Observe that, for each vertex  $v_i$  in  $G$ , there is a corresponding vertex  $u_i$  and a support vertex  $u'_i$  in  $G'$  (for example, see the vertices  $u_1$  and  $u'_1$  in Fig. 1 (b) corresponding to the vertex  $v_1$  in Fig. 1 (a)).



**Fig. 1.** Construction of a planar bipartite graph  $G'$  from a planar graph  $G$ .

Again for each edge  $v_i v_j \in E$  in  $G$ , there are three vertices added in the corresponding edge  $u_i u_j$  in  $G'$ , i.e.,  $u_{ij}, u'_{ij}$ , and  $u''_{ij}$  (for example, see the three vertices added between  $u_1$  and  $u_2$  in Fig. 1 (b)). Observe that the extra vertices added in the graph  $G'$  do not affect the graph's planarity, and the odd cycles of the graph (if any) become even. Moreover, the degree of each vertex in the graph  $G'$  is at most four. Thus,  $G'$  is a planar bipartite graph with  $\Delta(G') \leq 4$ . If the total number of vertices and edges in graph  $G$  is  $n$  and  $m$ , respectively, then the number of vertices and edges in graph  $G'$  is  $|V'| = 2 \cdot n + 3 \cdot m$  and  $|E'| = n + 4 \cdot m$ . In a planar graph  $G = (V, E)$ ,  $|E| \leq 3 \cdot |V| - 6$ . So,  $|V'| < 11 \cdot n$  and  $|E'| < 13 \cdot n$ . Hence  $G'$  can be constructed in linear time.  $\square$

Now, we prove that 3PVC-PB is **NP-hard**. For the hardness proof, we show a linear time reduction from VC-PLA to 3PVC-PB. Let  $G = (V, E)$  be an instance of VC-PLA. Construct an instance  $G' = (V_1, V_2, E')$  of 3PVC-PB as discussed in Lemma 1. We prove the following claims to establish the **NP-hardness** result for the 3PVC-PB.

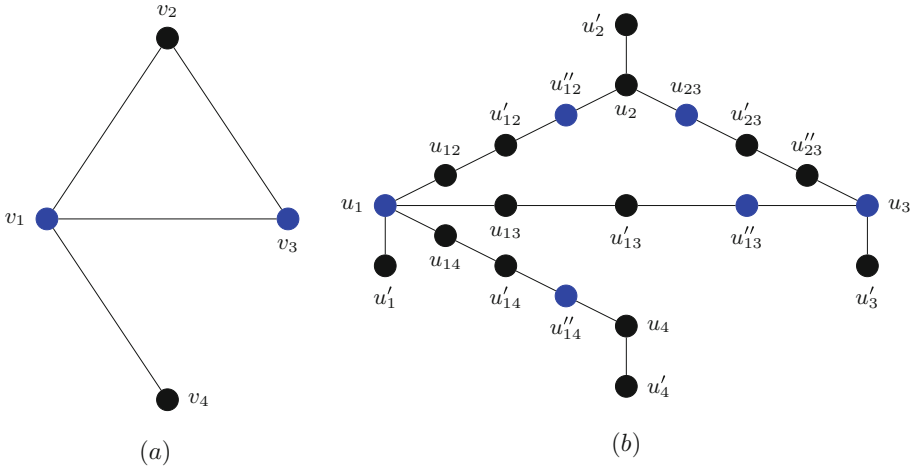
**Claim 1.** For each edge  $v_i v_j \in E$  in graph  $G$ , there exist three vertices  $u_{ij}, u'_{ij}$ , and  $u''_{ij}$  in graph  $G'$ . Out of these three vertices, one must be present in any 3-path vertex cover of graph  $G'$ .

*Proof.* The proof of the claim follows directly from the definition of the 3-path vertex cover. As the three vertices  $u_{ij}, u'_{ij}$ , and  $u''_{ij}$  form a path of order three, any 3-path vertex cover of the graph  $G'$  must contain at least one vertex out of these three vertices.  $\square$

**Claim 2.** For an edge  $v_i v_j \in E$  in  $G$ , if the corresponding vertices  $u_i$  and  $u_j$  of the edge in  $G'$  are not in a 3-path vertex cover set  $D$ , then either at least two vertices from the set  $\{u_{ij}, u'_{ij}, u''_{ij}\}$  or both the support vertices  $u'_i$  and  $u'_j$  are in  $D$ .

*Proof.* Assume that none of the vertices from the set  $\{u'_i, u_i, u_j, u'_j\}$  are in  $D$ . Moreover, there exists exactly one vertex from the set  $\{u_{ij}, u'_{ij}, u''_{ij}\}$  in  $D$  and  $D$  is a 3-path vertex cover in the graph  $G'$ . If  $u_{ij} \in D$ , then there are two paths of order three containing no vertices from  $D$ . The two paths are  $u'_j - u_j - u''_{ij}$  and  $u_j - u''_{ij} - u'_{ij}$ . It contradicts the fact that  $D$  is a 3-path vertex cover. If  $u'_{ij} \in D$ , then  $u'_i - u_i - u_{ij}$  and  $u'_j - u_j - u''_{ij}$  create two paths of order three, containing no vertices from  $D$ . It also contradicts the fact that  $D$  is a 3-path vertex cover. In the case that only  $u''_{ij} \in D$ , similar combinatorial arguments can be given to obtain a contradiction. Consider the case when neither of  $u_i$  and  $u_j$  belongs to  $D$ . Furthermore, we assume that  $D$  contains exactly one vertex from the set  $\{u_{ij}, u'_{ij}, u''_{ij}\}$ . In this case,  $D$  must contain both the support vertices  $u'_i$  and  $u'_j$  along with  $u''_{ij}$  in  $D$  to satisfy the condition of the 3-path vertex cover.

Therefore, it is proved that if both  $u_i$  and  $u_j$  are not in  $D$ , then either (i) at least two vertices from the set  $\{u_{ij}, u'_{ij}, u''_{ij}\}$  are in  $D$  or (ii) both the support vertices  $u'_i$  and  $u'_j$  must be in  $D$ . This proves the claim.  $\square$



**Fig. 2.** A 3PVC solution for  $G'$  constructed from a VC solution in  $G$ .

Now, we prove that 3PVC-PB is **NP-hard** by proving the following lemma.

**Lemma 2.**  $G$  has a vertex cover  $C$  with  $|C| \leq k$ , if and only if  $G'$  has a 3-path vertex cover  $D$  with  $|D| \leq k + m$ .

*Proof.* Let  $C \subseteq V$  be a vertex cover in the graph  $G = (V, E)$  having cardinality at most  $k$ . For each  $v_i \in C$ , take the corresponding vertices of  $v_i$  as  $u_i$  in the graph  $G' = (V', E')$ . Update  $D = D \cup \{u_i\}$ . After adding all the corresponding vertices of  $C$  in  $D$ ,  $|D| \leq k$ . As  $C$  is a vertex cover in  $G$ , for each edge  $v_i v_j \in E$ , either  $v_i$  or  $v_j$  must be in  $C$  (the tie can be broken by arbitrarily choosing one vertex if both the vertices are in  $C$ ). Without loss of generality, assume that  $v_i \in C$ . Take the corresponding vertex  $u_i$  in the graph  $G'$ . Now, add the 4<sup>th</sup>

vertex encountered in the path  $u_i \rightsquigarrow u_j$  in  $D$  (for example, see the path  $u_1 \rightsquigarrow u_2$  in Fig. 2 (b) corresponding to the edge  $v_1v_2$  in Fig. 2 (a). As  $v_1 \in C$ , the 4<sup>th</sup> vertex  $u''_{12}$  in the path  $u_1 \rightsquigarrow u_2$  is chosen in  $D$ ). Repeat the process for each edge in  $G$  and add one vertex to  $D$ . So, after completion of this step, the number of vertices added in  $D$  is  $m$  (number of edges present in  $G$ ). Now, observe that  $D$  is a 3-path vertex cover in  $G'$  as each path of order three contains at least one vertex from  $D$  and  $|D| \leq k + m$ .

Conversely, let  $D \subseteq V'$  be a 3-path vertex cover of size at most  $k + m$ . We argue that  $G$  has a vertex cover  $C$  of size at most  $k$ . Consider each vertex  $u_i \in D$  in  $G'$ . Take its corresponding vertex  $v_i$  in  $C$ . Clearly  $|C| \leq k$  (follows from Claim 1). If  $C$  is a vertex cover in  $G$ , then the proof completes. If  $C$  is not a vertex cover in  $G$ , then there must exist an edge  $v_iv_j \in E$  in  $G$ , such that  $v_i, v_j \notin C$ . Take the corresponding vertices of  $v_i$  and  $v_j$  as  $u_i$  and  $u_j$ , respectively, in  $G'$ . As  $v_i, v_j \notin C$ , the corresponding vertices  $u_i, u_j \notin D$ . That means  $D$  contains either at least two vertices from the set  $\{u_{ij}, u'_{ij}, u''_{ij}\}$  or both the support vertices  $u'_i$  and  $u'_j$  (follows from Claim 2). If both the support vertices  $u'_i$  and  $u'_j$  belong to  $D$ , then update  $D = D \cup \{u_i, u_j\} \setminus \{u'_i, u'_j\}$ . If the above condition fails, then there must exist two vertices from the set  $\{u_{ij}, u'_{ij}, u''_{ij}\}$  in  $D$ . Now, update  $D = D \cup \{u_i\} \setminus \{u_{ij}\}$ , if  $u_{ij} \in D$ , else update  $D = D \cup \{u_j\} \setminus \{u''_{ij}\}$ .

Update  $C$  and repeat the process till every edge in  $G$  has one of its end vertex in  $C$ . Due to Claim 1,  $C$  is a vertex cover having  $|C| \leq k$ . Therefore, 3PVC-PB is **NP-hard**. □

**Theorem 1.** 3PVC-PB is **NP-complete**

*Proof.* For a given set  $D \subseteq V$  in a bipartite graph  $G = (V, E)$  and a positive integer  $k$ , one can verify whether  $D$  is a 3PVC of size at most  $k$ . This can be done in linear time by checking whether there exists a path of order 3 in the subgraph induced by  $V \setminus D$ . Hence, the problem 3PVC-PB is in **NP**. As per Lemma 2, 3PVC-PB is **NP-hard**. Therefore, 3PVC-PB is **NP-complete**.

### 4 Approximation Algorithm

In this section, we design a 1.5-approximation algorithm for the 3PVC problem in bipartite graphs having maximum degree four. The proposed algorithm is a greedy algorithm, which runs in linear time. Note that, for the 3PVC problem, there exists a 1.57-approximation algorithm in cubic graphs [13] and a 2-approximation algorithm in general graphs [14, 15]. Let  $\psi_3(G)$  denote the cardinality of a minimum 3-path vertex cover set in  $G$ . Observe that if a graph  $G$  is a path or a cycle, then the following lemma for  $\psi_3(G)$  is valid.

**Lemma 3.** Let  $P_n$  denote a simple path on  $n$  vertices and  $C_n$  denote a cycle on  $n$  vertices, then  $\psi_3(P_n) = \lfloor \frac{n}{3} \rfloor \leq \frac{n}{3}$  and  $\psi_3(C_n) = \lceil \frac{n}{3} \rceil \leq \frac{n+2}{3}$ .

*Proof.* Consider a simple path  $P_n$  on  $n$  vertices. For  $i = 1, 2, \dots, \frac{n}{3}$ , select every 3-<sup>i</sup>th vertex from the path  $P_n$  in a set  $D$ . Observe that  $D$  is a 3PVC for the path  $P_n$  and  $|D| \leq \frac{n}{3}$ . In the case of a cycle  $C_n$ , a similar combinatorial argument can be given to prove that  $\psi_3(C_n) \leq \frac{n+2}{3}$ . □



Now, we discuss the algorithm to get a 3PVC set  $D$  in a bipartite graph  $G = (V, E)$ , where  $V = V_1 \cup V_2$ . The algorithm sequentially checks for all the vertices of  $G$ . When it encounters a vertex  $v \in V$  having degree three, it checks for the neighbors of  $v$ . Let  $\lambda$  denote the number of vertices in  $N(v)$  having degree at least three. If  $\lambda < 2$ , then the algorithm adds  $v$  in  $D$  and updates  $G$  by removing  $v$  from  $G$ . If  $\lambda \geq 2$ , then the algorithm computes the optimal solution (say  $D'$ ) in the subgraph induced by  $N[v]$ , and the neighbors of  $N(v)$ . Let  $V'$  be the set containing the vertices in  $N[v]$  and the neighbors of  $N(v)$ . As the input graph  $G$  is a bipartite graph with  $\Delta(G) \leq 4$ , there does not exist an edge between any pair of vertices in  $N(v)$  and  $|V'| \leq 13$ . Now, the algorithm updates  $D = D \cup D'$  and removes the vertices selected in  $D'$  from  $G$ .

When the algorithm encounters a vertex  $v \in V$  having degree four, it calculates the value of  $\lambda$  in  $N(v)$ . If  $\lambda < 3$ , then the algorithm adds  $v$  in  $D$  and updates  $G$  by removing  $v$  from  $G$ . If  $\lambda \geq 3$ , then the algorithm computes the optimal solution (say  $D'$ ) in the subgraph induced by  $N[v]$ , and the neighbors of  $N(v)$ . Let  $V'$  be the set containing the vertices in  $N[v]$  and the neighbors of  $N(v)$ . Observe that,  $|V'| \leq 17$ . Now, the algorithm updates  $D = D \cup D'$  and removes the vertices selected in  $D'$  from  $G$ .

The algorithm continues the above procedure for each vertex in  $G$ . At last, when the graph contains only paths and cycles, it optimally computes the solution and adds it to  $D$ .

The above steps are summarized in Algorithm 1.

**Lemma 4.** *The set  $D$ , returned by Algorithm 1, is a 3PVC for the given bipartite graph  $G$ .*

*Proof.* The algorithm sequentially checks for all the vertices of  $G$ . For each vertex  $v \in V$  encountered with a degree three, the algorithm checks for its neighbors. If less than two neighbors have degree at least three, the algorithm adds the vertex  $v$  in  $D$  and removes  $v$  from  $G$ . If there are at least two neighbors of  $v$  having degree at least three, then the algorithm finds the optimal solution for the subgraph induced by  $N[v]$  and neighbors of  $N(v)$ . Then, it adds the optimal solution obtained from this induced subgraph in  $D$  and removes the vertices of the optimal solution from  $G$ . When the algorithm encounters a vertex  $v \in V$  of degree four, it checks for its neighbors. If less than three vertices in  $N(v)$  have degree at least three, it adds the vertex  $v$  in  $D$  and removes  $v$  from  $G$ . Otherwise, it finds the optimal solution in the subgraph induced by  $N[v]$  and neighbors of  $N(v)$ . Then, it adds the optimal solution obtained from this induced subgraph in  $D$  and removes the vertices of the optimal solution from  $G$ . The algorithm repeats the process in the updated graph  $G'$  until only paths and/or cycles remain in  $G'$ . Note that the set  $D'$  is the minimum 3-path vertex cover of  $G'$  (follows from Lemma 3). Thus,  $D = D \cup D'$  returned by Algorithm 1 is a 3-path vertex cover for the given bipartite graph  $G$ .  $\square$

**Lemma 5.** *Algorithm 1 runs in time linear in the number of vertices of the input graph, in the worst case.*

**Algorithm 1.** 3PVC-PB**Require:** A bipartite graph  $G = (V, E)$  of maximum degree 4.**Ensure:** A 3PVC set  $D$  of  $G$ .

---

```

1:  $D \leftarrow \emptyset$ ,  $G' \leftarrow G$ , and  $V' \leftarrow V$ .
2: for every  $v \in V'$  do
3:   if  $d_{G'}(v) = 3$  then
4:     Let  $\lambda$  denote the number of vertices in  $N(v)$  having a degree of at least 3.
5:     if  $\lambda \leq 1$  then
6:        $D \leftarrow D \cup \{v\}$ .
7:        $G' \leftarrow G' \setminus \{v\}$ ,  $V' \leftarrow V' \setminus \{v\}$ .
8:     else
9:       Let  $G'' = (V'', E'')$  be a graph, where  $V''$  consists of  $N[v]$  and the
neighbors of  $N(v)$ .
10:      Find a minimum 3PVC set  $D'$  in  $G''$ .
11:      Update  $D \leftarrow D \cup D'$ .
12:       $G' \leftarrow G' \setminus D'$ ,  $V' \leftarrow V' \setminus D'$ .
13:    else
14:      if  $d_{G'}(v) = 4$  then
15:        Let  $\lambda$  denote the number of vertices in  $N(v)$  having degree at least 3.
16:        if  $\lambda \leq 2$  then
17:           $D \leftarrow D \cup \{v\}$ .
18:           $G' \leftarrow G' \setminus \{v\}$ ,  $V' \leftarrow V' \setminus \{v\}$ .
19:        else
20:          Let  $G'' = (V'', E'')$  be a graph, where  $V''$  consists of  $N[v]$  and the
neighbors of  $N(v)$ .
21:          Find a minimum 3PVC set  $D'$  in  $G''$ .
22:          Update  $D \leftarrow D \cup D'$ .
23:           $G' \leftarrow G' \setminus D'$ ,  $V' \leftarrow V' \setminus D'$ .
24: Find a minimum 3-path vertex cover set  $D'$  of  $G'$ .            $\triangleright$  follows from Lemma 3
25: Update  $D \leftarrow D \cup D'$ .
26: return  $D$ .
```

---

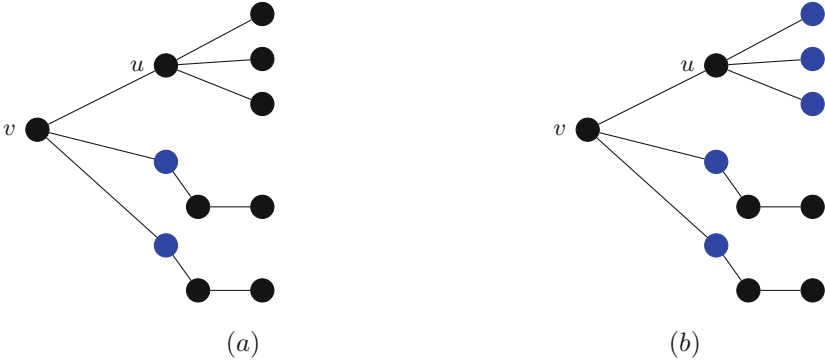
*Proof.* Observe that Algorithm 1 checks all the vertices sequentially in step 2. If the number of vertices in the input graph  $G$  is  $n$ , then this step will take  $O(n)$  time. For each vertex  $v \in V$  with  $d_G(v) \geq 3$ , the algorithm computes  $\lambda$ , the number of vertices in  $N(v)$  having degree at least three. Based on  $\lambda$ , the algorithm computes the solution for the subgraph induced by  $N[v]$  and neighbors of  $N(v)$ . These steps of the algorithm take constant time. Again, Algorithm 1 finds a 3-path vertex cover in the updated graph  $G'$  in step 24. This step can be computed in  $O(n)$  time as  $G'$  consists of only paths and cycles. All the other steps of the algorithm take constant time. Hence, the worst-case time taken by Algorithm 1 is  $O(n)$ .  $\square$

**Lemma 6.** Let  $D$  be a 3PVC set returned by Algorithm 1 and  $OPT$  be a 3PVC set of minimum size for the given bipartite graph  $G$ , then  $|D| \leq \frac{3}{2} \cdot |OPT|$ .

*Proof.* Observe that Algorithm 1 evaluates a 3-path vertex cover of minimum size for the cycles and paths (see step 24 of the algorithm). So, for proving the

approximation factor, we consider the vertices with degree at least three taken in  $D$  and prove that  $\frac{|D|}{|OPT|} \leq \frac{3}{2}$ .

Consider each vertex  $v \in D$  of degree at least three. If  $v \in OPT$ , then the algorithm achieves the best-case scenario by including  $v$  in  $D$ . Without loss of generality, assume that  $v \notin OPT$ . If  $d_G(v) = 3$ , the algorithm considers two cases to add  $v$  in  $D$ .

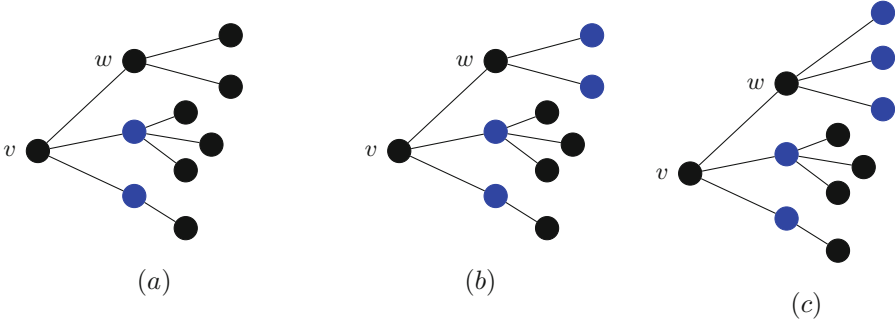


**Fig. 3.** Instance of a degree 3 vertex having exactly one neighbor with degree at least 3.

The first case, considered by the algorithm, is when there exists at most one vertex  $u \in N(v)$  such that  $d_G(u) \geq 3$  (see steps 5–7 in Algorithm 1). As  $v \notin OPT$ , at least two vertices from  $N(v)$  must be in  $OPT$  to make the  $OPT$  a 3PVC in  $G$  (for example, see Fig. 3 (a)). If the algorithm chooses two vertices from  $N(v)$  along with  $v$  in  $D$ , then the approximation factor is  $\frac{3}{2}$ . If the algorithm chooses all the three vertices of  $N(v)$ , then either  $OPT$  contains  $N(v)$  or  $u \notin OPT$ . In the former case, the approximation factor is  $\frac{4}{3} < \frac{3}{2}$ . Note that the two vertices (say  $x$  and  $y$ ) in  $N(v) \setminus \{u\}$  has degree at most two. The algorithm selects  $v$  in  $D$  and removes  $v$  from  $G$ . Now, the degree of the two vertices  $x$  and  $y$  is one, and still, the algorithm includes them in  $D$ , which means both the vertices  $x$  and  $y$  are in  $OPT$ .

In the latter case, as both  $u, v \notin OPT$ ,  $N(u)$  must be in  $OPT$  (for example, see Fig. 3 (b)). For the worst-case scenario, assume that  $N(u) \subseteq D$ . In that case,  $d_G(u) = 4$ . If  $d_G(u) = 3$ , then after removing the vertex  $v$  from  $G$ , makes the degree of  $u$  as two. The algorithm computes the optimal solution for all the degree one and two vertices at last. There is no way the algorithm selects  $N[u]$  optimally. If the degree of the vertices of  $N(u)$  is at least three and selected by the algorithm earlier, then after removing  $N(u)$ , the degree of  $u$  becomes 0 (cannot be in  $D$ ). So, the only way the algorithm includes  $N[u]$  in  $D$ , if  $d_G(u) = 4$ .

Now, we argue the approximation factor of the algorithm by considering both the vertices  $u$  ( $d_G(u) = 4$ ) and  $v$  ( $d_G(v) = 3$ ). Observe that  $OPT$  contains at least five vertices from  $N(u)$  and  $N(v)$  (see Fig. 3 (b)), whereas  $D$  contains at most seven vertices, including both  $u$  and  $v$ . So, the approximation factor is  $\frac{7}{5} < \frac{3}{2}$ .



**Fig. 4.** Instance of a degree 3 vertex with at least two neighbors of degree at least 3.

For  $d_G(v) = 3$ , the second case considered by the algorithm is when there exist at least two vertices  $w, u \in N(v)$  such that  $d_G(w) \geq 3$  and  $d_G(u) \geq 3$  (see steps 9–12 in Algorithm 1). In the second case, the algorithm finds an optimal solution for the subgraph induced by  $N[v]$  and neighbors of  $N(v)$ . As  $v \notin OPT$ , at least two vertices from  $N(v)$  must be in  $OPT$ . If the algorithm computes three vertices in  $D$  from the induced subgraph, then the approximation factor is  $\frac{3}{2}$ .

Consider the case that the algorithm chooses all the four vertices of  $N[v]$  in  $D$ , i.e.,  $N[v] \subseteq D$ . Assume that  $OPT$  consists of two vertices from  $N(v)$ , i.e., there exists a vertex  $w \in N(v)$ , such that  $w \notin OPT$  (for example, see Fig. 4 (a)). As both  $v$  and  $w$  are not in  $OPT$ , there must be the case that  $N(w) \subseteq OPT$  (for example, see Fig. 4 (b)). As the algorithm chooses  $w$  in  $D$ ,  $d_G(w) \geq 3$ . Otherwise, the algorithm would not choose  $w$  in  $D$  as already  $v \in D$ , and the algorithm computes the optimal solution in the subgraph induced by  $N[v]$  and neighbors of  $N(v)$ . Now, we argue the approximation factor of the algorithm by considering both the vertices  $v$  ( $d_G(v) = 3$ ) and  $w$ . Observe that  $OPT$  contains at least two vertices from  $N(v)$  and  $N(w)$ , whereas  $D$  contains  $N[v]$  and  $N(w)$ . If  $d_G(w) = 3$ , the approximation factor is  $\frac{6}{4}$ . If  $d_G(w) = 4$ , the approximation factor is  $\frac{7}{5}$  (for example, see Fig. 4 (c)). So, for  $d_G(v) = 3$ , the approximation factor is at most  $\frac{3}{2}$ . Observe that, for each vertex  $v \in D$  with  $d_G(v) = 4$ , the algorithm considers two cases to include  $v$  in  $D$ . Out of these two cases, the first case deals with the scenario when at most two vertices in  $N(v)$  have a degree of at least three. The second case deals with the scenario when at least three vertices in  $N(v)$  have a degree of at least three. The rest of the proof follows the similar combinatorial arguments given for the degree three vertices above. Thus, for each possible case, the approximation factor is at most  $\frac{3}{2}$ . Therefore, the solution  $D$ , returned by the algorithm, is  $\frac{3}{2} \cdot |OPT|$ , i.e.,  $|D| \leq \frac{3}{2} \cdot |OPT|$ .  $\square$

**Theorem 2.** Algorithm 1 is a  $\frac{3}{2}$ -approximation algorithm for the 3-path vertex cover problem in bipartite graph  $G$  with maximum degree four. The algorithm runs in  $O(n)$  time.

*Proof.* Follows from Lemma 4, Lemma 5, and Lemma 6. □

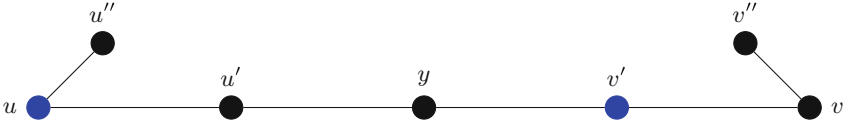
## 5 Approximation Complexity

In this section, we show that the 3-path vertex cover (3PVC) problem is **APX-complete** in bipartite graphs by exhibiting an **L-reduction** [4] from the vertex cover problem in cubic graphs (VC-CG) to the 3PVC problem in bipartite graphs (3PVC-BP). Note that VC-CG is known to be **APX-complete** [4].

**Lemma 7** [3]. *If  $G = (V, E)$  is a cubic graph and  $C_{opt}$  is a minimum vertex cover in  $G$ , then  $|C_{opt}| \geq \frac{|V|}{2}$ .*

**Construction:** Let cubic graph  $G = (V, E)$  denote an instance of VC-CG. We construct an instance of 3PVC-BP (bipartite graph  $G' = (V_1, V_2, E')$ ) as follows:

Replace each edge  $uv \in E$  of  $G$  by a path  $u \rightsquigarrow v$  of five vertices in  $G'$ , where the end vertices of the path are  $u$  and  $v$  (see Fig. 5). We call these three vertices added in the path  $u \rightsquigarrow v$  other than  $u$  and  $v$  as *added* vertices. We also call the end vertices  $u$  and  $v$  as *node* vertices. For each vertex  $v_i \in V$  in the graph  $G$ , add a support vertex, say  $v_i''$  and an edge  $vv_i''$  in the graph  $G'$  (for example, see the edges  $uu''$  and  $vv''$  in Fig. 5). Note that the construction is similar to the **NP-hardness** proof construction of Sect. 3 (see Fig. 1). From this construction, it follows that  $|V'| = 2 \cdot |V| + 3 \cdot |E| = 2 \cdot |V| + \frac{3 \cdot |V|}{2} \cdot 3 < 7 \cdot |V|$  and  $|E'| = 4 \cdot |E| + |V| = 4 \cdot \frac{3 \cdot |V|}{2} + |V| = 7 \cdot |V|$ .



**Fig. 5.** Gadget for an edge of the graph  $G$ .

To prove that 3PVC-BP is **APX-complete**, we first prove that 3PVC-BP is **APX-hard**. The **APX-hardness** is proved by reducing the VC-CG to the 3PVC-BP via an **L-reduction**. Let  $G = (V, E)$  be an instance of VC-CG. Construct the instance  $G' = (V', E')$  of 3PVC-BP as discussed above.

**Lemma 8.** *3PVC-BP is APX-hard.*

*Proof.* Let  $C \subseteq V$  be a vertex cover (VC) in the graph  $G = (V, E)$ . We construct a 3-path vertex cover  $D$  for the graph  $G' = (V', E')$  from  $C$  as follows:

For each vertex  $v_i \in C$  in  $G$ , take the corresponding vertex  $v_i$  in  $D$  from  $G'$ . As  $C$  is a VC in  $G$ , for each edge  $uv \in E$  in  $G$ , at least one of the end vertices of the edge  $uv$  must be in  $C$ . If  $u \in C$ , then include the added vertex  $v'$  from  $G'$  in  $D$ ; otherwise, include the added vertex  $u'$  in  $D$ .

Now, we prove that  $D$  is a 3PVC in  $G'$ . Observe that, for each edge-gadget corresponding to each edge  $uv$  in  $G$ ,  $D$  contains either the vertices  $u$  and  $v'$ , if  $u \in C$  or  $v$  and  $u'$ . So, from every edge-gadget, the vertices selected in  $D$  forbid a 3-path with no vertices in  $D$ . Further, if one of the vertices among  $u$  and  $v$  is not in  $D$ , then its two adjacent vertices from the other two edge-gadgets must be in  $D$ . Therefore, the subgraph induced by the vertices  $V' \setminus D$  does not have a path of order three. Thus,  $D$  is a 3PVC for the graph  $G'$ .

Let the number of vertices in the graph  $G$  be  $n$ , i.e.,  $|V| = n$ . As  $G$  is a cubic graph,  $|E| = \frac{3 \cdot n}{2}$ . There is precisely one added vertex taken in  $D$  from each edge-gadget along with the vertices in  $C$ . Therefore,  $|D| = |C| + |E| = |C| + \frac{3 \cdot n}{2}$ . Let  $C_{opt}$  be a minimum vertex cover for  $G$ , then by Lemma 7,  $|C_{opt}| \geq \frac{n}{2}$ . Let  $D_{opt}$  be a minimum 3PVC for  $G'$ , then  $|D_{opt}| \leq |C_{opt}| + 3 \cdot \frac{n}{2} \leq |C_{opt}| + 3 \cdot |C_{opt}| \leq 4 \cdot |C_{opt}|$ .

Conversely, let  $D$  be a 3-path vertex cover in  $G'$ . If  $D$  contains any of the support vertices, we delete the support vertex and add its neighbor (node vertex) in  $D$  if the neighbor is not in  $D$ . We construct a vertex cover  $C$  from the 3-path vertex cover  $D$  as follows:

For each node vertices  $v_i \in D$  in  $G'$ , take its corresponding vertex from  $G$  in  $C$ . If  $C$  is a VC in  $G$ , then the proof is complete. If  $C$  is not a VC in  $G$ , then there exists an edge  $uv \in E$  in  $G$ , for which neither of the end vertices is in  $C$ . Observe that, in the corresponding edge-gadget of the edge  $uv$ ,  $D$  does not contain the node vertices and the support vertices (otherwise, one of the vertices  $u$  or  $v$  must be in  $C$ ). That means  $D$  contains the added vertices  $u'$  (otherwise,  $u'' - u - u'$  form a 3-path) and  $v'$  (otherwise,  $v'' - v - v'$  form a 3-path) to satisfy the 3PVC condition. We remove the vertex  $u'$  from  $D$  and add  $u$  in  $D$ . Repeat the process until getting a VC in the graph  $G$ .

Observe that  $|C| \leq |D| - |E| \leq |D| - 3 \cdot \frac{n}{2}$  (as from each edge-gadget, at least one vertex out of three added vertices must be in  $D$ ). Let  $D$  be any 3PVC of  $G'$  and  $C$  be a corresponding VC for  $G$ , and  $D_{opt}, C_{opt}$  be the minimum 3PVC for  $G'$  and corresponding minimum VC for  $G$ , respectively.

Then  $|D| - |D_{opt}| \geq |C| + 3 \cdot \frac{n}{2} - |C_{opt}| - 3 \cdot \frac{n}{2}$ .

$|D| - |D_{opt}| \geq |C| - |C_{opt}|$ .

This gives an **L**-reduction from VC-CG to 3PVC-BP with  $\alpha = 4$  and  $\beta = 1$ .  $\square$

Note that 3PVC-BP is in **APX** [9]. As per Lemma 8, 3PVC-BP is **APX-hard**. Therefore, 3PVC-BP is **APX-complete**.

## 6 Conclusion

In this paper, we studied the 3-path vertex cover problem in different graph classes. We provided a linear time **NP-completeness** proof for the problem in planar bipartite graphs. With respect to approximation algorithms, we proposed a 1.5-approximation algorithm for the 3PVC problem in linear time. We proved that the 3PVC problem is **APX-complete** in bipartite graphs.

From our perspective, the following open problems are worth pursuing:



1. An exact exponential algorithm [5] for the 3PVC problem in bipartite graphs - Note that such algorithms exist for general graphs, but we hope to exploit the bipartite structure to obtain more efficient algorithms.
2. Unit disk graphs [2] - It is well-known that the 3PVC problem is **NP-hard** in unit disk graphs. We plan to investigate non-trivial approximation algorithms for the same.

## References

1. Brešar, B., Kardoš, F., Katrenič, J., Semanišin, G.: Minimum k-path vertex cover. *Discrete Appl. Math.* **159**(12), 1189–1195 (2011)
2. Clark, B.N., Colbourn, C.J., Johnson, D.S.: Unit disk graphs. *Discrete Math.* **86**(1–3), 165–177 (1990)
3. Devi, N.S., Mane, A.C., Mishra, S.: Computational complexity of minimum P4 vertex cover problem for regular and K1, 4-free graphs. *Discrete Appl. Math.* **184**, 114–121 (2015)
4. Du, D., Ko, K.-I., Hu, X., et al.: *Design and Analysis of Approximation Algorithms*, vol. 62. Springer, New York (2012). <https://doi.org/10.1007/978-1-4614-1701-9>
5. Fomin, F.V., Kratsch, D.: *Exact Exponential Algorithms*. TTCSAES, Springer, Heidelberg (2010). <https://doi.org/10.1007/978-3-642-16533-7>
6. Gollmann, D.: Protocol analysis for concrete environments. In: Moreno Díaz, R., Pichler, F., Quesada Arencibia, A. (eds.) *EUROCAST 2005*. LNCS, vol. 3643, pp. 365–372. Springer, Heidelberg (2005). [https://doi.org/10.1007/11556985\\_47](https://doi.org/10.1007/11556985_47)
7. Kardoš, F., Katrenič, J., Schiermeyer, I.: On computing the minimum 3-path vertex cover and dissociation number of graphs. *Theoret. Comput. Sci.* **412**(50), 7009–7017 (2011)
8. Karp, R.M.: Reducibility among combinatorial problems. In: *Complexity of Computer Computations*, pp. 85–103. Plenum Press (1972)
9. Kumar, M., Mishra, S., Devi, N.S., Saurabh, S.: Approximation algorithms for node deletion problems on bipartite graphs with finite forbidden subgraph characterization. *Theoret. Comput. Sci.* **526**, 90–96 (2014)
10. Lichtenstein, D.: Planar formulae and their uses. *SIAM J. Comput.* **11**(2), 329–343 (1982)
11. Novotný, M.: Design and analysis of a generalized canvas protocol. In: *IFIP International Workshop on Information Security Theory and Practices*, pp. 106–121 (2010)
12. Tu, J., Shi, Y.: An efficient polynomial time approximation scheme for the vertex cover  $p_3$  problem on planar graphs. *Discussiones Math. Graph Theory* **39**(1), 2019
13. Jianhua, T., Yang, F.: The vertex cover P3 problem in cubic graphs. *Inf. Process. Lett.* **113**(13), 481–485 (2013)
14. Jianhua, T., Zhou, W.: A factor 2 approximation algorithm for the vertex cover P3 problem. *Inf. Process. Lett.* **111**(14), 683–686 (2011)
15. Jianhua, T., Zhou, W.: A primal-dual approximation algorithm for the vertex cover P3 problem. *Theoret. Comput. Sci.* **412**(50), 7044–7048 (2011)
16. Zhang, A., Chen, Y., Chen, Z.-Z., Lin, G.: Improved approximation algorithms for path vertex covers in regular graphs. *Algorithmica* **82**(10), 3041–3064 (2020)



# Competition-Based Generalized Self-profit Maximization in Dual-Attribute Networks

Liman Du<sup>(✉)</sup> , Wenguo Yang<sup></sup>, and Suixiang Gao

School of Mathematical Sciences, University of Chinese Academy of Science,  
Beijing 100049, China

duliman18@mails.ucas.edu.cn, {yangwg, sxgao}@ucas.ac.cn

**Abstract.** The purpose of Profit Maximization (PM) problem in social advertising is gaining profit generated by adopters as much as possible by launching some initial adopters as information sources to trigger the spread of promotion information. Many related studies mainly focus on pure network, single product and one-dimension diffusion model. Inspired by real advertising activities in social networks, we propose the Dual-Attribute Compete (DAC) model where the information about competitive entities can spread simultaneously. Moreover, it not only captures attributes of both potential consumers and products but also reflects the relationship between them. Under DAC model, we study the Competition-based Generalized Self-profit Maximization (CGSM) problem aiming to select at most  $k$  individuals to form an optimal seed set as the source of information diffusion, so as to maximize the profit obtained by marketers. Considering that the objective function of CGSM problem is generally nonsubmodular, we design R-CGSM algorithm. Based on martingale analysis and the concept of shapley value, it uses sandwich method to get a pretty good solution of CGSM problem. We evaluate our proposed algorithm by conducting experiments on one synthetic data set and three real world data sets. Results of experiments validate the effectiveness and accuracy of R-CGSM algorithm.

**Keywords:** Profit maximization · Nonsubmodularity · Competitive social advertising · Dual-Attribute networks

## 1 Introduction

Due to the rapid development and expansion of social media sites and platforms, numerous work, such as [2, 8, 18–21], focus on several extensions of classical Influence Maximization (IM) problem proposed in [12]. For example, Profit Maximization (PM) problem proposed in [19, 20] changes the objective function from influence spread to profit spread. [3, 4] pay attention to the similar problem

Supported by the National Natural Science Foundation of China under grant numbers 12071459 and 11991022 and the Fundamental Research Funds for the Central Universities under Grant Number E1E40107X2.



with nonsubmodular objective function. The main difference between them is that the former confines the number of seed nodes while the latter underlines the influence of nodes' similarity on their interaction. Considering that many algorithms designed for IM problem requires the objective function to be submodular, they are not suitable for nonsubmodular PM problem. [5, 6, 9–11, 15, 22] provide new methods to solve nonsubmodular optimization problem. Except for changing objective function, some extensions of IM problem are proposed under novel diffusion models. In order to capture the full spectrum of entity interactions from competition to complementarity, [14] proposes Com-IC model and studies IM problem under this model. Multi-Feature diffusion model is firstly proposed in [7]. Then, Multi-attribute Independent Cascade Model and Multi-attribute based Influence Maximization Problem are studied in [17] while budgeted profit maximization under multiple features propagation of one product is proposed in [1].

Inspired by competitive advertising, we propose Dual-Attribute Compete (DAC) model in this paper. It not only reflects the relationship between potential consumers and promotional products but also considers the influence of their attributes. Then we formulate the Competition-based Generalized Self-profit Maximization (CGSM) problem whose objective function is not submodular in general. Based on sandwich method, a three-phase algorithm, R-CGSM algorithm, making full of reverse reachable set with shapely value is designed in this paper. Its efficiency is evaluated by the results of a series of experiments conducted on both synthetic and real data sets of social network.

The remainder of this paper is organized as follows: Sect. 2 introduces Dual-Attribute Compete model and formulates CGSM problem. The detail of R-CGSM algorithm is given in Sect. 3. Section 4 shows the experimental results, and we conclude in Sect. 5.

## 2 Problem Formulation

### 2.1 Dual-Attribute Compete Model and Diffusion Dynamics

We abstract a Dual-Attribute network as  $G = (V, E, W, C, M, L, \Omega, \omega)$  where  $V$ ,  $E$ ,  $W$ ,  $C$ ,  $M$ ,  $L$ ,  $\Omega$  and  $\omega$  represent users, social ties, initial influence probability, non-overlapping community structure, users' emotion tendency, users' labels, the relationship between users and products' features and the weight of products' features, respectively. What should be emphasized is that community structure  $C = \{C_1, C_2, \dots\}$  satisfies  $V = \bigcup_{i=1}^{|C|} C_i$  and  $C_i \cap C_j = \emptyset$  for  $i, j = 1, 2, \dots, |C|$ . Now, we introduce the meaning of  $M, L, \Omega, \omega$ . For each node  $v \in V$  representing user of social networks,  $M_v = 0$  means node  $v$  is impartial and objective;  $M_v \in [-1, 0)$  if node  $v$  dislikes the promoted products or the company which launches this product, otherwise  $M_v \in (0, 1]$ . The binary-valued matrix  $L$  shows whether associations between nodes and labels exist or not. Given  $l$  labels,  $L_{v,i} = 1$  if node  $v$  holds label  $i$  for  $i = 1, 2, \dots, l$ , otherwise  $L_{v,i} = 0$ . These two kinds of information actually reveal user's attributes. For promoted products, sale-points can be regarded as their features. If node  $v$  is not interested in a sale-point remarked as  $i$ , the information about this feature can not persuade  $v$  to buy

the product. We set  $\Omega_{v,i} = 0$  to reflect this case. Therefore, given  $o$  features of promoted products,  $\Omega_{v,i} = 0$  or  $1$  for  $v \in V, i \in \{1, 2, \dots, o\}$ . At the same time,  $\sum_{i=1}^o \omega_i = 1$  because  $\omega$  is the weight of product's feature. Then, we recall the definition of influence probability proposed in [4].

**Definition 1.** ([4]) *The influence probability between adjacent nodes  $u$  and  $v$  in an attribute network is defined as  $P_{u,v} = a \cdot Pe_{u,v} + (1 - a) \cdot Pi_{u,v}$  where weight parameter  $a \in [0, 1]$ ,  $Pe$  and  $Pi$  are two kinds of influence strength which are based on initial influence probability  $W$  and calculated from the perspective of emotion tendency and social interactions.*

Inspired by the MF-model studied in [7] and Com-IC model proposed in [14], we define DAC model as follows. There are at least two entities involved in DAC model, and we focus on a special case where only two entities,  $\mathcal{A}$  and  $\mathcal{B}$ , are considered. Let  $\phi_{\mathcal{A}}(v) \in (0, 1]$  represent the modified profit with respect to entity  $\mathcal{A}$  generated by  $\mathcal{A}$ -adopter  $v$ .  $G' = \cup_{i=1}^o G^i$  and  $G^i = (V^i, E^i)$  is a sub-graph of  $G$ . Each node  $v \in V$  satisfying  $\Omega_{v,i} = 1$  is remarked as  $v^i \in V^i$  and  $E^i \subseteq E$ . For  $(u^i, v^i) \in E^i, i = 1, \dots, o$ ,  $p_{u^i, v^i} = P_{u,v}$  represents the probability with which user  $u^i$  successfully spreads the information about feature  $i$  to user  $v^i$ . Only the information about feature  $i$  spread on  $G^i$  and the process is independent to others. Now, we consider the information dissemination on  $G^i$ . We use  $v$  instead of  $v^i$  and set  $q^i(v) = q(v) = \{q_{\mathcal{A}|\emptyset}(v), q_{\mathcal{A}|\mathcal{B}}(v), q_{\mathcal{B}|\emptyset}(v), q_{\mathcal{B}|\mathcal{A}}(v)\}$ . All the nodes initially stay in the joint state of ( $\mathcal{A}^i$ -idle,  $\mathcal{B}^i$ -idle). For node  $v$  that does not accept  $\mathcal{B}^i$ , it transforms from  $\mathcal{A}^i$ -idle to  $\mathcal{A}^i$ -accepted with probability  $q_{\mathcal{A}|\emptyset}(v)$ . If  $v$  is  $\mathcal{B}^i$ -accepted and informed of  $\mathcal{A}^i$ , it becomes  $\mathcal{A}^i$ -accepted with probability  $q_{\mathcal{A}|\mathcal{B}}(v)$ . The meaning of  $q_{\mathcal{B}|\emptyset}(v)$  and  $q_{\mathcal{B}|\mathcal{A}}(v)$  are similar. We use the assumption that  $0 \leq q_{\mathcal{A}|\mathcal{B}}(v) < q_{\mathcal{A}|\emptyset}(v) = 1$  and  $0 \leq q_{\mathcal{B}|\mathcal{A}}(v) < q_{\mathcal{B}|\emptyset}(v) = 1$  for  $v \in V$  to reflect the competition between  $\mathcal{A}$  and  $\mathcal{B}$ . After all dissemination terminates, we check the adoption of node  $v$ . For  $v \in V$ , we define  $r(v, \mathcal{A}^i) = 1$  if node  $v$  is  $\mathcal{A}^i$ -accepted, otherwise  $r(v, \mathcal{A}^i) = 0$ . Each node  $v$  randomly picks an activation thresholds  $\theta_{\mathcal{A}}(v) \in [0, 1]$ . And if node  $v$  satisfying  $\sum_{i=1}^o \omega_i r(v, \mathcal{A}^i) \geq \theta_{\mathcal{A}}(v)$ , node  $v$  adopts  $\mathcal{A}$ . The activation condition of  $v$  with respect to  $\mathcal{B}$  is similar. Because of  $q_{\mathcal{A}|\mathcal{B}}(v) > 0$  and  $q_{\mathcal{B}|\mathcal{A}}(v) > 0$ , node  $v$  may accept both  $\mathcal{A}^i$  and  $\mathcal{B}^i$ .

Now, we consider the DAC model as a information diffusion model. Let  $S_{\mathcal{A}}, S_{\mathcal{B}} \subset V$  be two seed sets. At time  $t = 0$ , for  $i = 1, 2, \dots, o$ , each node belongs to  $S_{\mathcal{A}} \cap V^i$  accepts  $\mathcal{A}^i$  while  $\mathcal{B}^i$  is accepted by  $S_{\mathcal{B}} \cap V^i$ . At each time step  $t \geq 1$ , for a node  $u^i$  becoming  $\mathcal{A}^i$ -accepted at time  $t - 1$  and one of its neighbor  $v^i$ , information about  $\mathcal{A}^i$  has only one chance to successfully spread from  $u^i$  to  $v^i$  with probability  $P_{u,v}$ . If node  $v^i$  stays in the joint state of ( $\mathcal{A}^i$ -idle,  $\mathcal{B}^i$ -idle) and is informed about both  $\mathcal{A}^i$  and  $\mathcal{B}^i$  at the same time step, tie-breaking rule is used to decide its state. The tie-breaking rule consists of two phases: generate a random permutation  $\pi$  of  $v^i$ 's in-neighbors who point to  $v^i$  with live edges and accept  $\mathcal{A}^i$  or  $\mathcal{B}^i$ ; test  $v^i$  with each such in-neighbor and its accepted item following the order of  $\pi$ . If there is an in-neighbor  $u^i$  accepting both  $\mathcal{A}^i$  and  $\mathcal{B}^i$ , then test both of them following their order of acceptance by  $u^i$ . When the diffusion of all features are terminated, each node's adoption is fixed.

## 2.2 Problem Statements

Given that DAC model highlights the independent cascade diffusion of features, we underline some assumptions as follows before formulating Competition-based Generalized Self-profit Maximization problem.

- There exists no node satisfying  $\Omega_{v,i} = 0$  for every  $i \in \{1, 2, \dots, o\}$ , that is, each node  $v \in V$  is interested in or can accept at least one feature.
- If node  $v$  does not adopt  $\mathcal{A}$ , it can not generate profit with respect to  $\mathcal{A}$  regardless of its adoption about  $\mathcal{B}$ . What’s more, if  $v$  adopts both  $\mathcal{A}$  and  $\mathcal{B}$ , it still has chance to spread information to its neighbors whereas the profit with respect to  $\mathcal{A}$  generated by  $v$  does not be considered when the total profit is calculated.

**Definition 2 (CGSM problem).** *Given  $G = (V, E, W, C, M, L, \Omega, \omega)$ , constant  $k$ , two competitive entities  $\mathcal{A}$  and  $\mathcal{B}$ , parameter set  $q(v)$  and profit  $\phi_{\mathcal{A}}(v)$  for each node  $v \in V$ , coefficient  $a \in [0, 1]$  and probability distribution  $\mathbf{PB}(r)$  over  $\mathbf{B} = \{S_{\mathcal{B}} | S_{\mathcal{B}} \subset V, |S_{\mathcal{B}}| = r\}$  which collects all possible  $\mathcal{B}$  seed sets, Competition-based Generalized Self-profit Maximization problem aims to find a  $\mathcal{A}$ -seed set  $S_{\mathcal{A}}^* \subset V$  such that the expected profit generated by  $\mathcal{A}$ -adopters is maximized under DAC model, i.e.*

$$S_{\mathcal{A}}^* \in \arg \max_{S_{\mathcal{A}} \subseteq V, |S_{\mathcal{A}}| \leq k} \Phi_{\mathcal{A}}(S_{\mathcal{A}}, S_{\mathcal{B}}), \quad (1)$$

where  $\Phi_{\mathcal{A}}(S_{\mathcal{A}}, S_{\mathcal{B}}) = \sum_{S_{\mathcal{B}} \in \mathbf{B}} \Pr[S_{\mathcal{B}}] \sum_{i=1}^o \sum_{g^i} \Pr[g^i] \cdot \omega_i \sum_{v \in I_{g^i, S_{\mathcal{B}}}(S_{\mathcal{A}})} \phi_{\mathcal{A}}(v)$  and

$I_{g^i, S_{\mathcal{B}}}(S_{\mathcal{A}})$  is the set of nodes in the realization  $g^i$  with  $S_{\mathcal{B}}$  of subgraph  $G^i$  which can receive the information spread from  $S_{\mathcal{A}}$  and become ( $\mathcal{A}^i$ -accepted,  $\mathcal{B}^i$ -idle/rejected).

We propose the following theorem.

**Theorem 1.** *CGSM problem is NP-hard and computing the accurate value of  $\Phi_{\mathcal{A}}(S_{\mathcal{A}}, S_{\mathcal{B}})$  for two fixed seed sets  $S_{\mathcal{A}}$  and  $S_{\mathcal{B}}$  is #-P hard. Additionally, even though  $S_{\mathcal{B}}$  is fixed,  $\Phi_{\mathcal{A}}(S_{\mathcal{A}}, S_{\mathcal{B}})$  is not submodular with respect to  $S_{\mathcal{A}}$  in general.*

## 3 The Algorithm

To address CGSM problem, we propose R-CGSM algorithm which consists of three phases.

### 3.1 Model Influence Probability

Taking initial influence probability, community structure and nodes’ attributes into consideration, this phase recalculates the influence probability between nodes by Algorithm 1, where  $W_{u,v}$  is related to  $(u, v) \in E$  and  $W_u$  is a vector recording the initial influence probability between  $u$  and each of its neighbors.

**Algorithm 1.** Model Influence Probability ( $G = (V, E, W, C, M, L), a$ )

---

```

1: Initialize  $P, Pe, Pi, IS, TS, LS$  as zero matrix.
2: for  $(u, v) \in E$  do
3:   if  $M(u) \cdot M(v) > 0$  then
4:      $Pe_{u,v} = 1 - |M(u)| - |M(v)|$ .
5:   else
6:      $Pe_{u,v} = ||M(u)| - |M(v)||$ .
7:    $IS_{u,v} = \frac{W_{u,v} - \min_{u,v \in V} W_{u,v}}{\max_{u,v \in V} W_{u,v} - \min_{u,v \in V} W_{u,v}}$ .
8:   if  $u$  and  $v$  are in the same community then
9:      $TS_{u,v} = \sigma(W_u^T \cdot W_v)$  and  $LS_{u,v} = \sigma(L_u^T \cdot L_v)$  where  $\sigma(x) = (1 + e^{-x})^{-1}$ .
10:   $Pi_{u,v} = \left(1 + e^{6-4(IS_{u,v}+TS_{u,v}+LS_{u,v})}\right)^{-1}$ .
11:   $P_{u,v} = a \cdot Pe_{u,v} + (1 - a) \cdot Pi_{u,v}$ .
12: Return  $P$ 

```

---

### 3.2 Find Candidate Solutions

Given that the objective function of CGSM problem is nonsubmodular, it firstly produces approximate solutions for  $\Phi_{\mathcal{A},u}$  and  $\Phi_{\mathcal{A},l}$  representing the upper and lower bounds of CGSM problem's objective function  $\Phi_{\mathcal{A}}$ . Then, it finds a solution of  $\Phi_{\mathcal{A}}$  based on the concept of shapley value.

We have emphasized some assumptions in Sect. 2.2. Now,  $\Phi_{\mathcal{A},u}$  and  $\Phi_{\mathcal{A},l}$  are obtained by changing them. The former can be acquired by assuming profit with respect to  $\mathcal{A}$  generated by node  $v$  adopting both  $\mathcal{A}$  and  $\mathcal{B}$  does not be ignored. And the lower bound function  $\Phi_{\mathcal{A},l}$  is obtained under another assumption that the information spread terminates at an adopter of both  $\mathcal{A}$  and  $\mathcal{B}$ . Take a further step of Theorem 11 in [14], we can draw a conclusion that these two functions are submodular. Therefore, we can obtain solutions of  $\Phi_{\mathcal{A},l}$  and  $\Phi_{\mathcal{A},u}$  by Revised-IMM algorithm.

Now, we briefly introduce the framework of Revised-IMM algorithm. Given  $G = (V, E)$  and  $S_{\mathcal{B}}$  sampled according to  $P_{\mathcal{B}}(r)$ , randomly select  $v^i$  with probability  $\frac{\phi_{\mathcal{A}}(v) \cdot \omega_i}{\phi_{\mathcal{A}}(V)}$ . Generate a realization  $g^i$  of  $G^i$  by removing each  $(u, v) \in E^i$  with probability  $1 - p_{u,v}$  and get a reverse reachable set  $RM$  of  $v^i$ . Revised-IMM algorithm generates enough Multi-Sampling  $RM$  and puts them into  $\mathcal{RP}$  until a certain stopping condition is satisfied. Then, it uses greedy method to obtain  $S_{\mathcal{A},l}$  ( $S_{\mathcal{A},u}$ ) which is a solution of bound function  $\Phi_{\mathcal{A},l}$  ( $\Phi_{\mathcal{A},u}$ ). What should be highlighted is that Revised-IMM algorithm has two different procedures to generate  $RM$  when focusing on  $\Phi_{\mathcal{A},l}$  and  $\Phi_{\mathcal{A},u}$ .

As for objective function, the fundamental idea to solve it is based on shapley value, a well-known concept from cooperative game theory. As shown in [16], it may outperform greedy method in the terms of the solution quality when objective function is not submodular. Because computing shapley value of nodes exactly is difficult, we use some sampling techniques to find its estimator. The goal of Algorithm 2 is computing the marginal contribution that each node makes to the diffusion process. The test mentioned in Line 10 is based on tie-breaking

rule introduced in Sect. 2.1. Algorithm 3 generates a rank list by sorting nodes in non-increasing order according to their shapley value. And the result obtained by Algorithm 3 lays a sound foundation for Algorithm 4. After initializing seed set  $S_{\mathcal{A}}$  as an empty set, Algorithm 4 tests each node in the order given by Algorithm 3 and selects some nodes which are not adjacent to selected nodes. If every node

---

**Algorithm 2.** Compute Marginal Gain ( $G = (V, E), P, \Omega, \omega, PB(r), q, \phi_{\mathcal{A}}$ )

---

- 1: Initialize  $S_{\mathcal{A},temp}$  as an empty set and  $MG$  as a n-tuple zero vector. Create two empty queues  $\mathcal{A}_{can}$  and  $\mathcal{B}_{can}$ .
  - 2: Sample  $S^{\mathcal{B}}$  according to the probability distribution  $PB(r)$  and mark each node  $v$  in  $S^{\mathcal{B}}$  as  $\mathcal{B}^i$ -accepted if  $\Omega_{v,i} = 1$  for  $i = 1, 2, \dots, o$ .
  - 3: **for**  $j = 1$  to  $|V|$  **do**
  - 4:     Put the first  $j$  nodes in  $\mathcal{A}_{\pi}$  into  $S_{\mathcal{A},temp}$ .
  - 5:     **for** every subgraph  $G^i$  **do**
  - 6:         **for** every  $v \in S_{\mathcal{A},temp} \cap V^i$  **do**
  - 7:             Put  $v$  into  $\mathcal{A}_{can}$  and mark  $v$  as  $\mathcal{A}^i$ -accepted.
  - 8:         **while**  $\mathcal{A}_{can}$  is not empty **do**
  - 9:             **for** every out-neighbor  $w$  of nodes in  $\mathcal{A}_{can}$  **do**
  - 10:                 Test  $w$  and denote its joint state with respect to both  $\mathcal{A}$  and  $\mathcal{B}$ .
  - 11:                 **if**  $w$  is  $\mathcal{A}^i$ -accepted **then**
  - 12:                     Put  $w$  into  $\mathcal{A}_{can}$ .
  - 13:     Calculate the total profit based on the first  $j$  nodes in  $\mathcal{A}_{\pi}$ . And take the difference between it and the total profit based on the first  $j - 1$  nodes in  $\mathcal{A}_{\pi}$  as the marginal gain of the  $j$ -th node in  $\mathcal{A}_{\pi}$ .
  - 14: Update  $MC$  according to each node's marginal gain.
  - 15: **Return**  $MC$
- 

**Algorithm 3.** Construct Rank List( $G = (V, E), P, \Omega, \omega, PB(r), q, \phi_{\mathcal{A}}$ )

---

- 1: Denote the number of repetitions and a randomly sampled set of permutations as  $mc$  and  $\Pi$ , then initialize  $RL$  as an empty list.
  - 2: For  $i = 1, 2, \dots, o$ , construct subgraph  $G^i$ .
  - 3: **for all** node  $v \in V$  **do**
  - 4:     Initialize each node's shapley value to 0.
  - 5: **for all**  $\mathcal{A}_{\pi} \in \Pi$  **do**
  - 6:     **for all** nodes in  $\mathcal{A}_{\pi}$  **do**
  - 7:         Initialize nodes' temporal shapley value to 0 and  $MG$  to a zero matrix.
  - 8:         **for**  $k = 1$  to  $mc$  **do**
  - 9:              $MG' =$  Compute Marginal Gain ( $G = (V, E), P, \Omega, \omega, PB(r), q, \phi_{\mathcal{A}}$ ).
  - 10:              $MG = MG + MG'$ .
  - 11:         Update nodes' temporal shapley value as average of elements of  $MG$ .
  - 12:     Update nodes' shapley value as average value of their temporal shapley value.
  - 13: Sort the nodes in non-increasing order by their shapley value to obtain  $RL$ .
  - 14: **Return**  $RL$
-

has already been selected as seed or is adjacent to at least one seed node, we reconsider nodes in  $V \setminus S_{\mathcal{A}}$  and put one node whose shapley value is highest into the seed set.

---

**Algorithm 4.** Select nodes  $(RL, n, k)$ 


---

```

1: Initialize  $S_{\mathcal{A}}$  as an empty set and  $S_{can}$  as an empty list.
2: for  $i = 1$  to  $n$  do
3:   Denote the  $i$ -th element of  $RL$  as  $v$ .
4:   if  $v \notin S_{\mathcal{A}}$  and  $v$  is not adjacent to any node in  $S_{\mathcal{A}}$  then
5:     Put  $v$  into  $S_{\mathcal{A}}$ .
6:     if  $|S_{\mathcal{A}}| \geq k$  then
7:       Break.
8:   else
9:     Append  $v$  to  $S_{can}$ .
10: if  $|S_{\mathcal{A}}| < k$  then
11:   for  $j = 1$  to  $k - |S_{\mathcal{A}}|$  do
12:     Put the  $j$ -th element of  $S_{can}$  into  $S_{\mathcal{A}}$ .
13: Return  $S_{\mathcal{A}}$ 

```

---

### 3.3 Sandwich Approximation

---

**Algorithm 5.** R-CGSM

---

```

1:  $P =$  Model influence probability  $(G = (V, E, W, C, M, L), a)$ .
2: Initialize  $S_{\mathcal{A}, \Phi}$ ,  $S_{\mathcal{A}, \Phi_u}$  and  $S_{\mathcal{A}, \Phi_l}$  as empty sets.
3:  $RL =$  Construct Rank List  $(G = (V, E), P, \Omega, \omega, PB(r), q, \phi_{\mathcal{A}})$ .
4:  $S_{\mathcal{A}, \Phi} =$  Select nodes  $(RL, n = |V|, k)$ .
5:  $S_{\mathcal{A}, \Phi_u} =$  Reversed IMM  $(\Phi_{\mathcal{A}, u})$  and  $S_{\mathcal{A}, \Phi_l} =$  Reversed IMM  $(\Phi_{\mathcal{A}, l})$ .
6:  $S_{\mathcal{A}}^* = \arg \max_{S_{\mathcal{A}} \in \{S_{\mathcal{A}, \Phi}, S_{\mathcal{A}, \Phi_u}, S_{\mathcal{A}, \Phi_l}\}} \Phi_{\mathcal{A}}(S_{\mathcal{A}}, S_{\mathcal{B}})$ .
7: Return  $S_{\mathcal{A}}^*$ 

```

---

Algorithm 5 firstly finds solution  $S_{\mathcal{A}, \Phi}$ ,  $S_{\mathcal{A}, \Phi_l}$  and  $S_{\mathcal{A}, \Phi_u}$  for  $\Phi_{\mathcal{A}}(S_{\mathcal{A}}, S_{\mathcal{B}})$ ,  $\Phi_{\mathcal{A}, l}$  and  $\Phi_{\mathcal{A}, u}$ . Then, it selects the final optimal approximation solution  $S_{\mathcal{A}}^*$  among  $S_{\mathcal{A}, \Phi}$ ,  $S_{\mathcal{A}, \Phi_u}$  and  $S_{\mathcal{A}, \Phi_l}$  such that  $\Phi_{\mathcal{A}}(S_{\mathcal{A}}, S_{\mathcal{B}})$  reaches its maximum with  $S_{\mathcal{A}}^*$ .

**Theorem 2.**  $S_{\mathcal{A}}^*$  returned by R-CGSM algorithm satisfies

$$\Phi_{\mathcal{A}}(S_{\mathcal{A}}^*, S_{\mathcal{B}}) \geq \max \left\{ \frac{\Phi_{\mathcal{A}}(S_{\mathcal{A}, \Phi_u}, S_{\mathcal{B}})}{\Phi_{\mathcal{A}, u}(S_{\mathcal{A}, \Phi_u}, S_{\mathcal{B}})}, \frac{\Phi_{\mathcal{A}, l}(S_{\mathcal{A}^o}, S_{\mathcal{B}})}{\Phi_{\mathcal{A}}(S_{\mathcal{A}^o}, S_{\mathcal{B}})} \right\} \cdot (1 - 1/e) \cdot \Phi_{\mathcal{A}}(S_{\mathcal{A}}^o, S_{\mathcal{B}}),$$

where  $S_{\mathcal{A}}^o$  is the optimal solution maximizing  $\Phi_{\mathcal{A}}(S_{\mathcal{A}}, S_{\mathcal{B}})$ .

## 4 Experiment

A synthetic graph and three real-world social networks are used in our experiments. And the algorithms are implemented in Python.

- Synthetic: This is a relatively small acyclic directed graph randomly generated with 2708 nodes and 5278 edges.
- PFH(Petster-Friendships-Hamster data set in [13]): This network contains 12534 social relationships between 1858 users, abstracted from the website hamsterster.com.
- PHH(Petster-Hamster-Household data set in [13]): 921 nodes and 4032 edges are included in PHH data set, where nodes represent individuals and edges represent friendship between individuals.
- MI (Moreno-Innovation data set in [13]): This directed network captures innovation spread among 246 physicians. A node represents a physician and an edge between two physicians shows their friendship or cooperation.

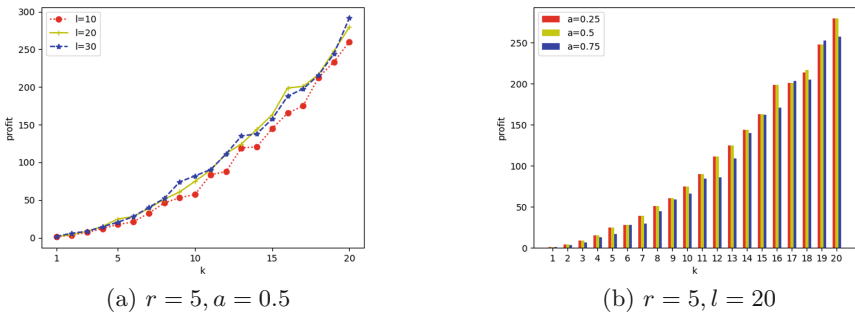


Fig. 1. influence of  $l, a$  on Synthetic.

Firstly, we conduct some experiments on Synthetic to study the influence of two parameters,  $l$  and  $a$ . Each element in  $W$  is chosen from  $\{0.1, 0.01, 0.001\}$  and the community structure is the result of Louvain Algorithm. Results are shown in Fig. 1. When a few seed nodes are needed, the difference of profit generated in different cases are not big. Although profit under the condition  $l = 10$  is always smallest, the gap of profit with  $l = 20$  and  $l = 30$  is not pretty large. In real social advertising, more details usually mean higher cost when collecting data. So we set  $l = 20$  for following experiments. The second one illustrates that the profit changes with weight  $a$ , which suggests marketers to pay attention to not only social interaction but also emotion tendency of potential adopters or purchasers in social networks.

Then, we concentrate on the influence of  $r$ . When calculating the profit generated by  $\mathcal{A}$ -adopters on Synthetic, PHH and MI data set, Fig. 2 reflects the influence of the number of seed picked by its competitive entity  $\mathcal{B}$ . We can draw

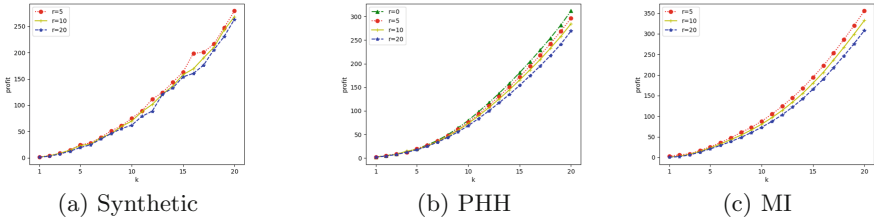


Fig. 2. influence of  $r$

a conclusion that less initial adopter hired by  $\mathcal{B}$  will lead to more profit for  $\mathcal{A}$ . Such a result is consistent with our common knowledge about competition.

Now, we study the influence of other parameters on Mi data set. After confining the number of seed nodes for  $\mathcal{B}$ , we explore the influence of  $o$ . As is shown in Fig. 3 (a), more information about feature of entities leads to smaller value of objective function. The reason may be that getting more information about two competitors can make potential consumer more likely to accept both of them. They are unsure of which one is better to choose. As a result, the expected profit decreases. If we relax the assumption such that the state of potential shopper with respect to  $\mathcal{B}$  does not impact the calculation of profit related to  $\mathcal{A}$ , such conclusion may change. Other figures in Fig. 3 show the influence of the choice

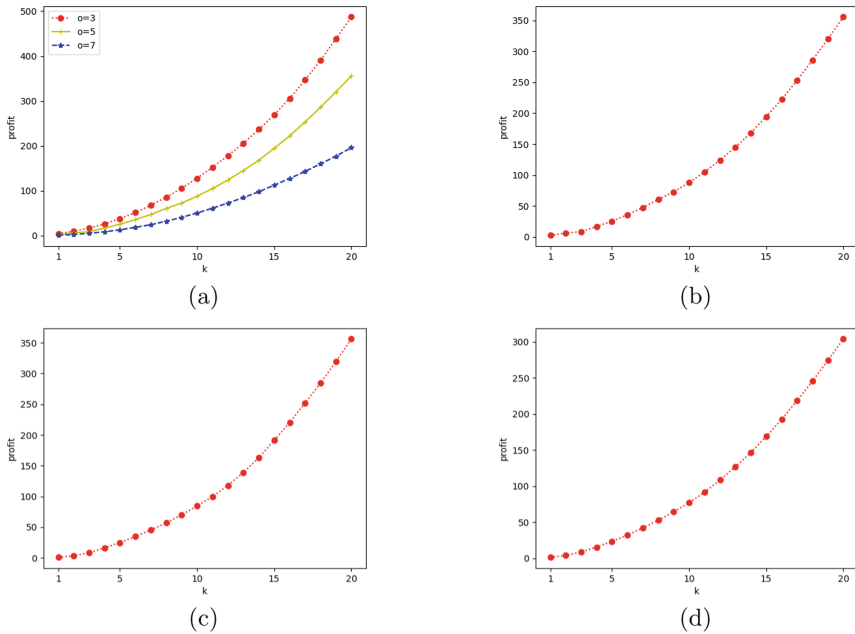
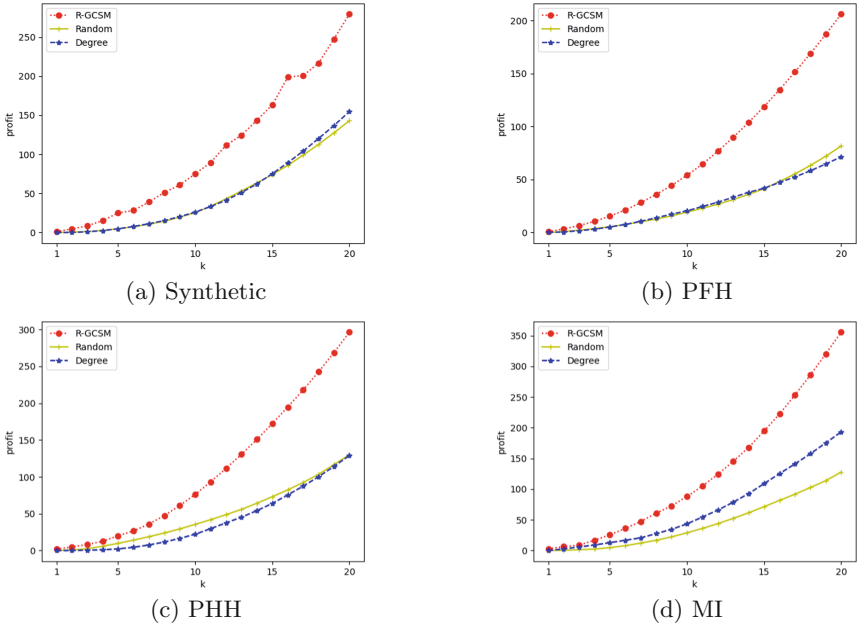


Fig. 3. influence of  $o$ ,  $W$  and  $C$  on MI



of  $W$  and  $C$ . We respectively conduct experiments under three assumptions: (b)  $W$  is randomly generated while  $|C| = |V|$ ; (c)  $|C| = |V|$  and each element in  $W$  satisfies  $w_{u,v} = \frac{1}{d(v)}$  where  $d(v)$  is the in-degree of node  $v$ ; (d) both  $W$  and  $C$  are randomly generated while  $|C| < |V|$ . Different choices result in different value of objective function despite the fact it always increases with  $k$ . Without special introduction, we conduct further experiments with the third setting.



**Fig. 4.** comparison of algorithms.

Last but not least, we compare R-CGSM with other two algorithms. One is Random algorithm which randomly selects seed nodes for  $\mathcal{A}$ . The other is Degree algorithm sorting all the nodes according to their degree and picks the top-20 nodes as seeds. Figure 4 illustrates the expected profit generated by all the  $\mathcal{A}$ -adopter on these three algorithms. The performance of algorithms is compared in four different data sets while  $k$  varies from 1 to 20. As is shown, R-CGSM outperforms other two algorithms.

## 5 Conclusion

In this paper, we propose Competition-based Generalized Self-profit Maximization problem in Dual-Attribute networks. Because the objective function of CGSM problem is generally nonsubmodular, we design R-CGSM to address it. R-CGSM algorithm is a three-phase algorithm and combines the concept

of shapley value and the method of sampling. Finally, a series of experiments are conducted on both artificial and real-world networks to evaluate the effectiveness of our proposed algorithm. The simulation results show the superiority of R-CGSM algorithm.

## References

1. Chen, T., Guo, J., Wu, W.: Adaptive multi-feature budgeted profit maximization in social networks (2020). <https://arxiv.org/abs/2006.03222>
2. Chen, W., Lin, T., Tan, Z., Zhao, M., Zhou, X.: Robust influence maximization. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 795–804. ACM, San Francisco California USA (2016). <https://doi.org/10.1145/2939672.2939745>
3. Du, L., Chen, S., Gao, S., Yang, W.: Nonsubmodular constrained profit maximization from increment perspective. *J. Comb. Optim.* <https://doi.org/10.1007/s10878-021-00774-6>
4. Du, L., Yang, W., Gao, S.: Generalized self-profit maximization in attribute networks. In: Du, D.-Z., Du, D., Wu, C., Xu, D. (eds.) COCOA 2021. LNCS, vol. 13135, pp. 333–347. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-92681-6\\_27](https://doi.org/10.1007/978-3-030-92681-6_27)
5. Feige, U., Izsak, R.: Welfare maximization and the supermodular degree. In: Proceedings of the 4th conference on Innovations in Theoretical Computer Science - ITCS '13, p. 247. ACM Press, Berkeley, California, USA (2013). <https://doi.org/10.1145/2422436.2422466>
6. Feldman, M., Izsak, R.: Constrained monotone function maximization and the supermodular degree (2014). <https://arxiv.org/abs/1407.6328v2>
7. Guo, J., Chen, T., Wu, W.: A multi-feature diffusion model: rumor blocking in social networks. In: IEEE/ACM Transactions on Networking, pp. 1–12 (2020). <https://doi.org/10.1109/TNET.2020.3032893>
8. He, X., Kempe, D.: Stability of influence maximization. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '14, pp. 1256–1265. ACM Press, New York, New York, USA (2014). <https://doi.org/10.1145/2623330.2623746>
9. Iyer, R., Jegelka, S., Bilmes, J.: Curvature and optimal algorithms for learning and minimizing submodular functions (2013). <https://arxiv.org/abs/1311.2110>
10. Iyer, R.K., Bilmes, J.A.: Submodular optimization with submodular cover and submodular knapsack constraints (2013). <https://arxiv.org/abs/1311.2106>
11. Iyer, R.K., Bilmes, J.A.: Algorithms for approximate minimization of the difference between submodular functions, with applications (2014). <https://arxiv.org/abs/1408.2051>
12. Kempe, D., Kleinberg, J., Tardos, E.: Maximizing the spread of influence through a social network. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 37–146 (2003). <https://doi.org/10.1145/956750.956769>
13. Kunegis, J.: KONECT - The Koblenz network collection. In: Proceedings of the International Conference on World Wide Web Companion, pp. 1343–1350 (2013). <https://doi.org/10.1145/2487788.2488173>
14. Lu, W., Chen, W., Lakshmanan, L.V.S.: From competition to complementarity comparative influence diffusion and maximization. *Proc. VLDB Endow.* **9**(2), 60–71 (2015). <https://doi.org/10.14778/2850578.2850581>

15. Narasimhan, M., Bilmes, J.A.: A submodular-supermodular procedure with applications to discriminative structure learning (2012). <https://arxiv.org/abs/1207.1404>
16. Narayanam, R., Narahari, Y.: A shapley value-based approach to discover influential nodes in social networks. *IEEE Trans. Autom. Sci. Eng.* **8**(1), 130–147 (2011). <https://doi.org/10.1109/TASE.2010.2052042>
17. Ni, Q., Guo, J., Du, H.W.: Multi-attribute based influence maximization in social networks. In: Wu, W., Du, H. (eds.) *AAIM 2021*. LNCS, vol. 13153, pp. 240–251. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-93176-6\\_21](https://doi.org/10.1007/978-3-030-93176-6_21)
18. Shang, J., Zhou, S., Li, X., Liu, L., Wu, H.: Cofim: a community-based framework for influence maximization on large-scale networks. *Knowl. Based Syst.* **117**, 88–100 (2017). <https://doi.org/10.1016/j.knosys.2016.09.029>
19. Tang, J., Tang, X., Yuan, J.: Towards profit maximization for online social network providers (2017). <https://arxiv.org/abs/1712.08963>
20. Tang, J., Tang, X., Yuan, J.: Profit maximization for viral marketing in online social networks: algorithms and analysis. *IEEE Trans. Knowl. Data Eng.* **30**(6), 1095–1108 (2018). <https://doi.org/10.1109/TKDE.2017.2787757>
21. Wang, Z., Yang, Y., Pei, J., Chu, L., Chen, E.: Activity maximization by effective information diffusion in social networks. *IEEE Trans. Knowl. Data Eng.* **29**(11), 2374–2387 (2017). <https://doi.org/10.1109/TKDE.2017.2740284>
22. Wu, W.L., Zhang, Z., Du, D.Z.: Set function optimization. *J. Oper. Res. Soc. China* **7** (2018). <https://doi.org/10.1007/s40305-018-0233-3>



# Largest Convex Hulls for Constant Size, Convex-Hull Disjoint Clusters

Xuehou Tan<sup>1,2</sup>(✉)  and Rong Chen<sup>1</sup>

<sup>1</sup> Dalian Maritime University, Linghai Road 1, Dalian, China

<sup>2</sup> Tokai University, 4-1-1 Kitakaname, Hiratsuka 259-1292, Japan  
xtan@tsc.u-tokai.ac.jp

**Abstract.** A *cluster* is a set of points, with a predefined similarity measure. In this paper, we study the problem of computing the largest possible convex hulls, measured by length and by area, of the points that are selected from a set of *convex-hull disjoint* clusters, one per cluster. We show that the largest convex hulls for convex-hull disjoint clusters of *constant size*, measured by length or area, can be computed in  $O(n^4)$  time, where  $n$  is the sum of cardinalities of all clusters. Our solution of either considered problem is *doubly* founded on a structure of clusters, whose all points are in convex position. The restricted problem for the set of clusters, whose points are in convex position, can be reduced to a sequence of subproblems of computing the single-source shortest-paths in a weighted graph. Not only our results significantly improve upon the known time bound  $O(n^9)$ , but also our algorithms can be used to improve the known results on the problems of computing largest convex hulls for disjoint line segments or squares.

## 1 Introduction

Imprecision on input data has been studied for a couple of decades in computational geometry, because coordinates of the input points obtained from the real world may have some error interval [5]. An imprecise point is usually given by a polygonal region, in which the exact point lies [7]. Various methods for handling imprecise points have been proposed, when some computation is done, for example to find the convex hull [5], the Voronoi diagram [1], or a traveling sales route [4].

Löffler and van Kreveld [7] have given a systematic study of convex hulls for imprecise points. Assume that an imprecise point is represented by a convex region (e.g., a line segment or a disk), which contains the exact point. The convex hull for convex regions is defined as that of the points, which are selected from regions, one per region. Then, it is a natural requirement to compute the largest and/or smallest possible convex hulls, measured by length and by area. Most variants of this problem are NP-hard, except for few special cases in which the

---

The work by Tan was partially supported by JSPS KAKENHI Grant Number 20K11683.

input is a set of squares or parallel line segments [7]. Recently, Díaz-Báñez et al. [3] gave an important result that all four variants of the problem can be solved in polynomial-time, provided that convex regions are *pairwise disjoint*. However, the time complexities of their algorithms may be as high as  $O(n^9)$  [3].

Observe that the vertices of convex regions play an important role in many computations concerning imprecise points. In most cases, the computation of largest and smallest possible convex hulls for imprecise points needs only the vertices defining the regions; otherwise, most considered problems are NP-hard [3, 7]. For example, the known polynomial-time algorithms for computing the convex hulls for a set of disjoint convex regions, presented by Löffler and van Kreveld [7] and Díaz-Báñez et al. [3], rely on the fact that any considered problem has an optimum solution in which all vertices of the reported convex hull belong to the set of vertices of given regions. Also, it is clear that representing an imprecise point with the vertices of its corresponding region makes the considered problems be simpler, and it is suitable in real applications.

From the above discussion, the concept of clusters can be used to represent imprecise points. A *cluster* is a set of points, with a predefined similarity measure; the exact point is assumed to be one of points in the cluster. In this paper, we study the problems of computing the largest possible convex hulls, measured by length and by area, of the points that are selected from a set of constant size, *convex-hull disjoint* clusters, one per cluster.

## 1.1 Our Results

In this paper, we focus on the computation of convex hulls for a set  $S$  of constant size, convex-hull disjoint clusters, or simply, disjoint clusters of constant size. We will refer to *MaxPCH* and *MaxACH* as the problems of finding the convex hulls for  $S$ , which are of maximum perimeter and maximum area, respectively.

A solution to either considered problem is *feasible* if it is the convex hull of points selected from clusters, one per cluster. A solution is *optimum*, denoted by *OPT*, if its corresponding measure is maximum.

Let  $P$  be the set of points of all clusters, and denote by  $CH(P)$  the convex hull of all points of  $P$ . The first result for our algorithms is the following.

**Lemma 1.** *OPT is identical to  $CH(P)$ , or there exists a cluster  $C$  such that at least two points of  $C$  are on  $CH(P)$  and exactly one point of  $C$  belongs to OPT.*

**Proof.** Omitted in this extended abstract.  $\square$

A set of points is said to be in *convex position* if all points form the vertices of a convex polygon. An important observation made in this paper is that any restricted problem for a set of clusters, whose all points are in convex position, can be reduced to a number of the subproblems of computing the single-source shortest-paths in a weighted graph. For a set of arbitrarily given point clusters, the points on  $CH(P)$  also help divide the considered problem into subproblems, which deal with point clusters in convex position.

We will first present the  $O(n^4)$  time algorithms for *MaxPCH* and *MaxACH* for a set of disjoint clusters of size two. (A cluster of size two can be considered as the *discrete* representation of a line segment.) Both algorithms can be generalized to a set of disjoint clusters of constant size, with the same time bound. Not only our results significantly improve upon the known time bound  $O(n^9)$ , but also the obtained solutions are unified and simple. Moreover, our algorithms can be used to improve the known results on the largest convex hull problem for disjoint line segments or squares. A summary of our new results on *MaxPCH* and *MaxACH*, and some of their variants is given in Table 1.

**Table 1.** Summary of new and previous results on *MaxPCH* and *MaxACH*, and some of their variants.

Problem	Model	Restrictions	New results	Previous ones
MaxPCH, MaxACH	Clusters	Disjoint, constant size	$O(n^4)$ (Th. 4)	$O(n^9)$ [3]
MaxACH	Line segments	Disjoint, convex position	$O(n^2)$ (Th. 1)	$O(n^3)$ [7]
MaxACH	Squares	Non-overlapping	$O(n^4)$ (Th. 4)	$O(n^5)$ [6]
MaxPCH	Squares	Non-overlapping	$O(n^4)$ (Th. 4)	$O(n^{10})$ [7]

## 2 Algorithms for a Set of Disjoint Clusters of Size Two

Denote by  $S$  a set of disjoint clusters of size two, and  $OPT$  a largest convex hull for  $S$ , measured by perimeter or area. We use  $\{a, b\}$  or  $\{a', b'\}$  to represent the points of a cluster. For points  $p$  and  $q$  in the plane, denote by  $pq$  the line segment joining  $p$  and  $q$ , and  $|pq|$  the length of  $pq$ . Denote by  $|R|$  and  $\|R\|$  the perimeter length and area of a region  $R$ , respectively. If  $R'$  is a convex chain, we also use  $|R'|$  and  $\|R'\|$  to represent the perimeter length of  $R'$  and area of the region bounded by  $R'$  and the segment connecting two endpoints of  $R'$ , respectively.

In the following, we present our algorithm for the problem *MaxPCH* for a set of  $n$  disjoint clusters of size two, and then modify it slightly for *MaxACH*.

### 2.1 An Overview

Assume below that some clusters have both points  $a$  and  $b$  on  $CH(P)$ ; otherwise,  $CH(P)$  is just the solution  $OPT$  (Lemma 1). Consider first the situation in which that there exists a cluster such that one of its points, denoted by  $s$ , is on  $CH(P)$ , and the other is in the interior of  $CH(P)$ . From the definition of *MaxPCH*, point  $s$  is on  $OPT$ . Let  $t$  be the (identical) copy of point  $s$ . Take  $s$  and  $t$  as two points on  $CH(P)$ . Then, the problem *MaxPCH* is reduced to that of finding the maximum-perimeter convex hull of  $s$  and the selected points, one per other cluster. Let  $U \leftarrow CH(P)$ . Denote by  $MPCH(U, s, t)$  the restricted problem *MaxPCH*, in which  $s$  (resp.  $t$ ) is called the source (resp. target) point.

The other situation is that if a point  $a$  of any cluster is on  $CH(P)$ , point  $b$  is on  $CH(P)$ , too. Let us fix a cluster  $C = \{a, b\}$ . By letting  $s, t \leftarrow a$  and  $U \leftarrow CH(P - \{b\})$ , we face with a problem  $MPCH(U, s, t)$  of finding the maximum-perimeter convex hull of  $s$  and the selected points, one per other cluster. Also, by letting  $s, t \leftarrow b$  and  $U \leftarrow CH(P - \{a\})$ , we have the other problem  $MPCH(U, s, t)$ . Clearly,  $OPT$  is the largest of their solutions.

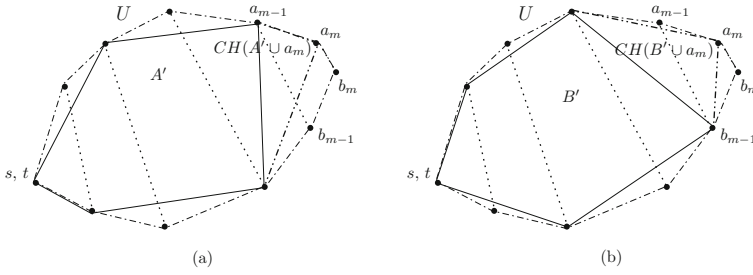
Let us focus on the optimum solution to  $MPCH(U, s, t)$ . If a cluster has only one of its points, say,  $p$  on chain  $U$ , then  $p$  appears in  $OPT$ . Such clusters can be easily dealt with, because  $OPT$  can be obtained from the solutions of  $MPCH(U_1, s, p)$  and  $MPCH(U_2, p, t)$ , where  $U_1$  is the portion of  $U$  between  $s$  and  $p$ , and  $U_2$  is the portion of  $U$  between  $p$  and  $t$ .

From the above discussion, we will consider only the clusters, whose points  $a$  and  $b$  are on  $U$ . Denote by  $T$  the set of clusters, whose points  $a$  and  $b$  are both on  $U$ . Consider a clockwise scan of points on  $U$ , starting from  $s$ . Suppose that for each cluster  $\{a, b\}$  in  $T$ , point  $a$  is encountered before  $b$ . From the cluster disjointness, we define an order of the clusters in  $T$  as that of their points  $a$  on chain  $U$ . Assume that the number of elements in  $T$  is  $m$ , i.e.,  $T = \{\{a_1, b_1\}, \dots, \{a_m, b_m\}\}$ . For ease of presentation, we will use  $T_i$  to denote the first  $i$  elements of  $T$ ,  $1 \leq i \leq m$ . Clearly, in addition to  $U$ ,  $s$  and  $t$ , set  $T$  is also an input of  $MPCH(U, s, t)$ .

Our method is to first solve the simplest variant of  $MPCH(U, s, t)$ , in which the points of all clusters are in convex position, and for each cluster in  $T$ , two points  $a_i$  and  $b_i$ ,  $1 \leq i \leq m - 1$ , are not adjacent on  $U$ , except for  $a_m$  and  $b_m$ . See Fig. 1 for an example, where the endpoints of a dotted segment represent a cluster of two points. Particularly, we call it an *atomic* problem  $MPCH(U, s, t)$ . A linear time solution to the atomic problem is given. Next, we present an  $O(n^2)$  time algorithm for a variant of  $MPCH(U, s, t)$ , in which the points of all clusters are in convex position. Finally, our solution is generalized to a set of arbitrarily given clusters.

## 2.2 Optimum Solution of an Atomic Problem

Denote by  $\mathcal{C}$  an optimum solution of the atomic problem  $MPCH(U, s, t)$ . For ease of presentation,  $\mathcal{C}$  is also called an *atomic chain*. A dynamic programming algorithm can be given. If  $m = 1$ , then  $\mathcal{C}$  is determined by the larger of  $(|sa| + |at|)$  and  $(|sb| + |bt|)$ . Assume that  $m > 1$ . Since either  $a_m$  or  $b_m$  appears in  $\mathcal{C}$ , we can consider two restricted solutions  $A$  and  $B$ , which contain  $a_m$  and  $b_m$  respectively. The solution for  $T_m$  (i.e.,  $|\mathcal{C}|$ ) is then the larger of  $|A|$  and  $|B|$ . For the atomic subproblem with respect to set  $T_{m-1}$ , let  $A'$  and  $B'$  be two restricted solutions such that  $A'$  and  $B'$  contain  $a_{m-1}$  and  $b_{m-1}$ , respectively (Fig. 1(a)). Denote by  $CH(a_m \cup A')$  (resp.  $CH(a_m \cup B')$ ) the convex hull of point  $a_m$  and all points of  $A'$  (resp.  $B'$ ). The larger of  $|CH(\{a_m\} \cup A')|$  and  $|CH(\{a_m\} \cup B')|$  is then  $|A|$ , and the larger of  $|CH(\{b_m\} \cup A')|$  and  $|CH(\{b_m\} \cup B')|$  is  $|B|$ . In this way,  $\mathcal{C}$  can be eventually computed.



**Fig. 1.** An instance of the atomic problem  $MPCH(U, s, t)$ .

Since all given points are in convex position, the operation of computing a convex hull, say,  $CH(\{a_m\} \cup A')$ , can be performed in constant time. This is because  $a_m$  is connected to two endpoints of a segment of  $A'$ , which has  $a_{m-1}$  as one of its endpoints. Hence,  $\mathcal{C}$  as well as  $|\mathcal{C}|$  can be found in linear time.

For the largest convex hull measured by *area*, if  $m = 1$ , then  $\mathcal{C}$  is the triangle of larger area, between  $\Delta_{s,a,t}$  and  $\Delta_{s,b,t}$ . (In the case that  $s$  is identical to  $t$ , the induction base is  $m = 2$ .) At every step of our dynamic programming algorithm, it also takes a constant time to find the triangle of larger area, between two considered triangles. Again,  $\mathcal{C}$  as well as  $\|\mathcal{C}\|$  can be found in linear time.

**Lemma 2.** *The atomic variant of  $MaxPCH$  or  $MaxACH$  for a set of disjoint clusters of size two can be solved in  $O(n)$  time, provided that  $CH(P)$  is given.*

### 2.3 Solving $MPCH(U, s, t)$ for Point Clusters in Convex Position

Let  $I$  be the (largest) set of clusters such that for any two consecutive clusters in  $I$ , say,  $\{a, b\}$  and  $\{a', b'\}$ , points  $b$  and  $a'$  are adjacent on  $U$ . Suppose that  $I$  has  $k$  clusters, say,  $C_1, C_2, \dots, C_k$ , indexed in the order in which the clusters appear on  $U$  (Fig. 2). For a cluster  $C_i = \{a, b\}$ ,  $1 \leq i \leq k$ , denote by  $J_i$  the set of clusters, whose points are on the portion of  $U$  between  $a$  and  $b$ , including  $C_i$  itself. Let  $J_0$  (resp.  $J_{k+1}$ ) be the set consisting of one cluster  $\{s, s\}$  (resp.  $\{t, t\}$ ).

Denote by  $|J_i|$  the number of clusters in  $J_i$ . Let  $\{a_{last}, b_{last}\}$  (resp.  $\{a_{first}, b_{first}\}$ ) be the last (resp. first) cluster of  $J_i$ ,  $1 \leq i \leq k$ . An atomic chain on  $J_i$  can be then defined as the optimum chain (in perimeter length or area) consisting of  $|J_i|$  points, one per cluster.

**Lemma 3.** *Any optimum solution  $OPT$  of  $MPCH(U, s, t)$  can be decomposed into a sequence of atomic chains, in which the connection between two consecutive chains consists of the line segments connecting point  $a_{last} \in J_i$  or  $b \in J_i$  with point  $a \in J_{i+1}$  or  $b_{last} \in J_{i+1}$ ,  $1 \leq i \leq k$ . Moreover, point  $s$  (resp.  $t$ ) is connected by a line segment to  $a \in J_1$  or  $b_{last} \in J_1$  (resp.  $b \in J_k$  or  $a_{last} \in J_k$ ).*

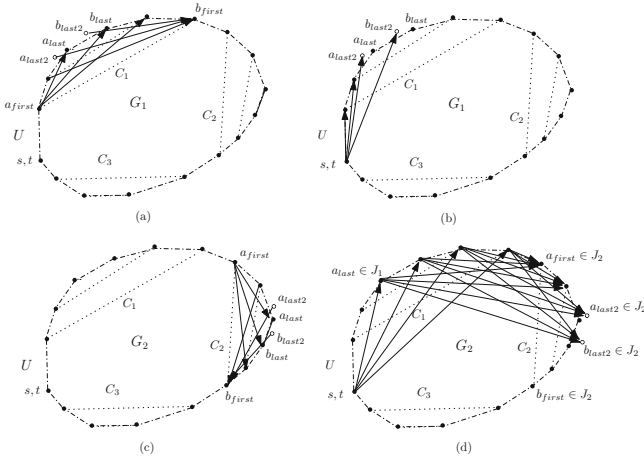
**Proof.** Omitted in this extended abstract.  $\square$



To solve the problem  $MPCH(U, s, t)$  for a set of point clusters in convex position, we will introduce a data structure  $G_i$  for each  $J_i$ , called the *atomic chain diagram* of  $J_i$ . The diagram  $G_i$  records all possible atomic chains defined on  $J_i$ , and a conjugation relation from  $J_{i-1}$  to  $J_i$ . Hence,  $k + 1$  atomic chain diagrams will be constructed. Finally, the Dijkstra paradigm is applied to atomic chain diagrams, so as to find  $OPT$ .

**Atomic Chain Diagrams.** From the (omitted) proof of Lemma 3, point  $s \in J_0$  may be connected by a line segment to point  $a \in J_1$  or point  $b_{last} \in J_1$ . Also, point  $b \in J_1$  or point  $a_{last} \in J_1$  may be connected to a point of  $J_2$ . To represent those connections correctly, we add copy of  $\{a_{last}, b_{last}\}$ , denoted by  $\{a_{last2}, b_{last2}\}$ , to  $J_1$ . See Figs. 2(a)-(b).

The first atomic chain diagram  $G_1$  is constructed as follows. First,  $s$  and all points  $a, b \in J_1$  (including  $a_{last2}, b_{last2} \in J_1$ ) are added to set  $V(G_1)$  as nodes  $s, a$  and  $b$ , respectively. Next, we add to  $E(G_1)$  (i) the arcs from point  $a_{first} \in J_1$  to  $a_{last} \in J_1$  and all points  $b \in J_1$ , excluding  $b_{last2} \in J_1$ , and the arcs from  $b_{last2} \in J_1$  and all points  $a \in J_1$ , excluding  $a_{first}, a_{last} \in J_1$ , to point  $b_{first} \in J_1$ . See Fig. 2(a). Clearly, the number of arcs of type (i) is  $2|J_1|$ . Also, we add to  $E(G_1)$  (ii) the arcs from  $s$  to  $b_{last2}$  and all points  $a \in J_1$ , excluding  $a_{last} \in J_1$ . See Fig. 2(b). The size of  $E(G_1)$  is thus  $O(|J_1|)$ .



**Fig. 2.** Illustration of constructing the diagram  $G_i$ .

In  $G_1$ , an arc  $(p, q)$  of type (i) represents an atomic problem  $MPCH(U', p, q)$  defined on  $J_1$ , where  $U'$  represents the portion of  $U$  from  $p$  to  $q$ . Notice that the atomic chain between  $a_{last} \in J_1$  (resp.  $b_{last} \in J_1$ ) and  $b_{first} \in J_1$  is given as the one between  $a_{last2}$  (resp.  $b_{last2}$ ) and  $b_{first}$ . Thus, the arcs of type (i) represent all possible atomic chains on  $J_1$ . The arcs of type (ii) represent the

connection from  $s$  to the starting points of atomic chains on  $J_1$ . Also,  $(s, a_{last2})$  and  $(s, b_{last2})$  represent the connections between  $s$  and  $a_{last}$  and between  $s$  and  $b_{last}$ , respectively. See Fig. 2(b). Therefore, all arcs of type (ii) together give a conjugation relation from  $s$  to atomic chains on  $J_1$ . We call  $s$  the *source node*, and  $a_{last} \in J_1$  and all points  $b \in J_1$  the *target nodes*, in  $G_1$ . Since  $a_{last2}$  (resp.  $b_{last2}$ ) is a copy of  $a_{last}$  (resp.  $b_{last}$ ), any path from  $s$  to a target node  $v$  in  $G_1$  consists of an arc from  $s$  to a node  $u$ , and an atomic chain between  $u$  and  $v$ .

The following diagram  $G_i$ ,  $2 \leq i \leq k+1$ , is constructed one by one. Again, add a copy of  $\{a_{last}, b_{last}\}$ , denoted by  $\{a_{last2}, b_{last2}\}$ , to  $J_i$ . Then,  $s, a_{last} \in J_{i-1}$ , all points  $b \in J_{i-1}$  and all points  $a, b \in J_i$  are added to set  $V(G_i)$  as nodes. For the last diagram  $G_{k+1}$ , the target node  $t$  is also added to  $V(G_{k+1})$ . Analogously, we add to  $E(G_i)$  (i) the arcs from point  $a_{first} \in J_i$  to  $a_{last} \in J_i$  and all points  $b \in J_i$ , excluding  $b_{last} \in J_i$ , and the arcs from  $b_{last2} \in J_i$  and all points  $a \in J_i$ , excluding  $a_{first}, a_{last} \in J_i$ , to point  $b_{first} \in J_i$ . See Fig. 2(c). Also, add to  $E(G_i)$  (ii) the arcs from  $a_{last} \in J_{i-1}$  and every point  $b \in J_{i-1}$  to  $b_{last2} \in J_i$  and all points  $a$  in  $J_i$ , excluding  $a_{last} \in J_i$ . Finally, add to  $E(G_i)$  (iii) the arcs from  $s$  to  $a_{last} \in J_{i-1}$  and all points  $b \in J_{i-1}$ , excluding  $b_{last2} \in J_{i-1}$ . See Fig. 2(d) for an instance of  $E(G_2)$ . Any  $sx$ -path in  $G_i$  thus consists of three arcs, one per type, in the order of (iii), (ii) and (i), except that no arcs of type (i) exist in  $G_{k+1}$ .

**The Algorithm.** We briefly review the Dijkstra paradigm [2]. Let  $G(V, E)$  be an directed graph such that each arc of  $E$  has a non-negative weight. The *length* of a path in  $G$  is the sum of the weights of its constituent arcs. The *shortest path* from node  $u \in V$  to node  $v \in V$  is then defined as any path from  $u$  to  $v$  with the minimum length among all possible paths, provided there exists a path from  $u$  to  $v$ . The so-called *single-source shortest-paths problem*, which can be solved using the Dijkstra's algorithm [2], asks to find a shortest path from a given source node  $s \in V$  to each node  $x \in V$ ,  $s \neq x$ .

In our application, all points are on chain  $U$ . To meet the requirement of finding a *longest* path from  $s$  to  $x$ , it suffices to define the weight of an  $sx$ -path  $W$  as  $(|U_{s,x}| - |W|)$ , where  $U_{s,x}$  denotes the portion of  $U$  from  $s$  to  $x$  clockwise, and  $|W|$  is the sum of Euclidean distances between the nodes of all arcs in  $W$ .

Let us now assign a weight with each arc of  $E(G_1)$ . The weight of an arc  $(x, y)$  of type (ii) is simply defined as  $|U_{x,y}| - |xy|$ . Since an arc  $(p, q)$  of type (i) defines an atomic subproblem on  $J_1$ , its weight is defined as the difference between  $|U_{p,q}|$  and the perimeter length, which is obtained by solving the atomic problem defined by  $p$  and  $q$  on  $J_1$ . Denote by  $GW_1$  the weighted diagram of  $G_1$ . From Lemma 3 as well as the definition of  $GW_1$ , the longest path from  $s$  to a target  $w \in J_1$  corresponds to the minimum weight path from  $s$  to  $w$  in  $GW_1$ .

For all other weighted diagrams  $GW_i$ ,  $2 \leq i \leq k+1$ , the arcs of types (i) and (ii) can be defined analogously. The weights of arcs of type (iii) in  $GW_i$  represent the optimum sub-solutions from  $s$  to  $a_{last} \in J_{i-1}$  and all points  $b \in J_{i-1}$ , in which the cluster input is limited to  $J_1 \cup J_2 \cup \dots \cup J_{i-1}$ . After  $GW_{i-1}$  is constructed, the weights of arcs of type (iii) in  $GW_i$  can be all computed and then assigned.

A path from the source node  $s$  to a target node  $x \in J_i$  in diagram  $GW_i$ ,  $2 \leq i \leq k + 1$ , is said to be *folded* in the sense that the first arc of type (iii) in the  $sx$ -path represents a (folded) path in  $GW_{i-1}$ . Any  $sx$ -path in  $GW_1$  is *not* folded, as it does not have any arc of type (iii).

**Lemma 4.** *Any optimum solution of  $MPCH(U, s, t)$  can be represented as a folded  $st$ -path in  $GW_{k+1}$ . Also, a folded  $st$ -path in  $GW_{k+1}$  corresponds to a feasible solution of  $MPCH(U, s, t)$ .*

**Proof.** Omitted in this extended abstract.  $\square$

**Lemma 5.** *Any diagram  $GW_i$ ,  $1 \leq i \leq k + 1$ , is acyclic.*

**Proof.** For any arc  $(p, q)$  in  $GW_i$ ,  $p$  is strictly prior to  $q$  on  $U$ . Moreover,  $s$  and  $t$  are two different nodes in  $GW_{k+1}$ . Hence, the lemma follows.  $\square$

**Lemma 6.** *The weighted diagrams  $GW_1, GW_2, \dots, GW_{k+1}$  can be all constructed in  $O(n^2)$  time and  $O(n^2)$  space.*

**Proof.** Omitted in this extended abstract.  $\square$

By now, we can give the first result of this paper.

**Theorem 1.** *Suppose that all given points are in convex position. The largest convex hull for a set of disjoint clusters of size two, measured by perimeter or area, can be computed in  $O(n^2)$  time.*

**Proof.** Omitted in this extended abstract.  $\square$

**Remark A.** The problem *MaxACH* for a set of non-intersecting line segments, whose endpoints are in convex position, has been studied; it is known that there exists an optimum solution in which all vertices of the reported convex hull are the vertices of segments [7]. Theorem 1 can be thus applied to it, improving upon the previously known  $O(n^3)$  time bound [7, Theorem 3].

## 2.4 Solving $MPCH(U, s, t)$ for Arbitrarily Given Points

In this section, the points of all clusters in  $S$  are arbitrarily given. Again, let  $T$  be the set of clusters, whose points  $a$  and  $b$  are on  $U$ . Our solution is given according to whether the clusters of  $T$  form a single or multiple atomic subproblems on  $U$ .

**The Clusters of  $T$  Form a Single Atomic Subproblem.** Assume that  $T$  has  $m$  clusters  $C_1, C_2, \dots, C_m$  such that for each cluster in  $T$ , its points  $a_i$  and  $b_i$ ,  $1 \leq i \leq m - 1$ , are not adjacent on  $U$  with respect to the clusters of  $T$ , except for  $a_m$  and  $b_m$ . Also, we use  $T_i$  to denote the first  $i$  elements of  $T$ ,  $1 \leq i \leq m$ .

Denote by  $\mathcal{D}$  an optimum solution of  $MPCH(U, s, t)$ . Again,  $MPCH(U, s, t)$  can be solved by a dynamic programming algorithm. Assume that  $m \geq 1$ . Since either  $a_m$  or  $b_m$  has to appear in  $\mathcal{D}$  (Lemma 1), we can consider two restricted, optimal solutions  $A$  and  $B$  such that  $A$  (resp.  $B$ ) contains  $a_m$  (resp.  $b_m$ ) and one point, per other cluster (of  $T$ ). The larger of  $|A|$  and  $|B|$  is then  $|\mathcal{D}|$ .

Consider now the subproblem with input  $T_{m-1}$ . Let  $A'$  and  $B'$  be two optimal solutions to the subproblem with input  $T_{m-1}$  such that  $A'$  and  $B'$  contain  $a_{m-1}$  and  $b_{m-1}$ , respectively. See Fig. 3(a). In order to obtain the solution, say,  $A$ , we first compute two convex hulls  $CH(\{a_m\} \cup A')$  and  $CH(\{a_m\} \cup B')$ . Here, a new difficulty is that some point clusters of  $S$ , which do not belong to  $T_m$  or  $T$ , may wholly be outside of the region bounded by  $CH(\{a_m\} \cup A')$  or  $CH(\{a_m\} \cup B')$ .

Let  $A_1$  be the optimal solution, which is computed from  $CH(\{a_m\} \cup A')$ . So,  $A_1$  is a candidate of  $A$ . Assume that some clusters in  $S$  are wholly outside of  $CH(\{a_m\} \cup A')$ ; otherwise,  $CH(\{a_m\} \cup A')$  gives  $A_1$ . Let  $b_i$  be the maximal point  $b$ , which belongs to  $T$  and appears in  $A'$ . By definition, segment  $a_{m-1}a_m$  and the portion of  $A'$  from  $b_i$  to  $a_{m-1}$  belong to  $A_1$  (Fig. 3(b)). The rest task is how to compute the convex chain of  $A_1$  from  $a_m$  to  $b_i$ .

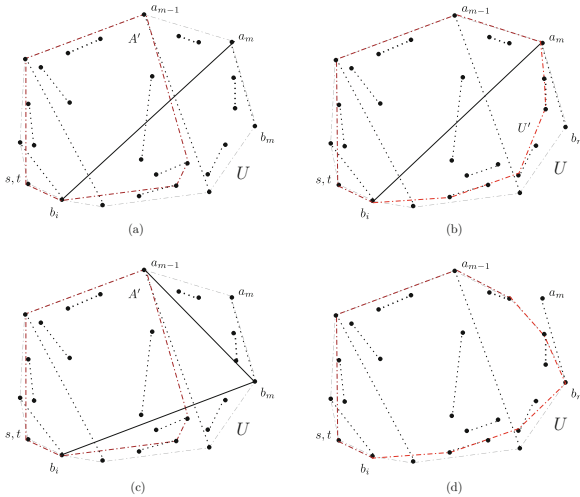


Fig. 3. Illustrating the computation of  $\mathcal{D}$ .

Let  $S_1(a_m)$  be the (non-empty) set of points, which lie in the interior of the region bounded by segment  $a_m b_i$  and the portion of  $U$  from  $a_m$  to  $b_i$  clockwise. Let us add  $a_m$  and  $b_i$  to  $S_1(a_m)$ , and then compute the convex hull  $CH(S_1(a_m))$ . Denote by  $U'$  be the obtained (convex) chain, excluding segment  $a_m b_i$ . The subproblem  $MPCH(U', a_m, b_i)$ , whose input is the set of clusters having at least one point on  $U'$ , can be solved using the algorithm presented in Sect. 2.3. The solution of  $MPCH(U', a_m, b_i)$ , called the *augmented segment connecting  $a_m$  and  $b_i$* , together with the portion of  $A'$  from  $b_j$  to  $a_m$  and segment  $a_{m-1}a_m$  then gives  $A_1$ , see Fig. 3(b). It thus takes  $O(|S_1(a_m)|^2)$  time to compute  $A_1$  (Theorem 1).

Consider now how to compute  $CH(\{b_m\} \cup A')$ , which produces a candidate  $B_1$  of  $B$ . Again, the portion of  $A'$  from  $b_i$  to  $a_{m-1}$  belongs to  $B_1$ . In this case, two convex chains of  $B_1$ , one from  $a_{m-1}$  to  $b_m$  and the other from  $b_m$  to  $b_i$ , are

needed to compute. This work is similar to the computation of the convex chain of  $A_1$  from  $a_m$  to  $b_i$ . See Figs. 3(c)-(d). It thus takes  $O(n^2)$  time to compute  $B_1$ . By symmetry, both  $CH(\{a_m\} \cup B')$  and  $CH(\{b_m\} \cup B')$  can be computed in  $O(n^2)$  time. The larger of  $A_1$  and  $A_2$  (which is computed from  $CH(\{a_m\} \cup B')$ ) is then  $A$ , and the larger of  $B_1$  and  $B_2$  (computed from  $CH(\{b_m\} \cup B')$ ) gives  $B$ . In this way,  $\mathcal{D}$  can be eventually obtained. We call the found solution  $\mathcal{D}$ , the *augmented atomic chain* from  $s$  to  $t$ .

Computing  $A$  and  $B$  from  $A'$  and  $B'$  takes  $O(n^2)$  time. Since the computation of  $A'$  and  $B'$  can be recursively done, the totally spent time is  $O(n^3)$ .

**Lemma 7.** *Suppose that  $S$  is a set of disjoint clusters of size two and the clusters of  $S$  on  $U$  form a single atomic subproblem. Then, the largest convex hull for  $S$ , measured by perimeter or area, can be computed in  $O(n^3)$  time*

**The Clusters of  $T$  Form Multiple Atomic Subproblems** As in Sect. 2.3, let  $I$  be the (largest) set of  $k$  clusters such that for any two consecutive clusters in  $I$ , say,  $\{a, b\}$  and  $\{a', b'\}$ , points  $b$  and  $a'$  are adjacent on  $U$ . Also, for a cluster  $C_i = \{a, b\}$  in  $I$  ( $1 \leq i \leq k$ ), denote by  $J_i$  the set of clusters, whose points are on the portion of  $U$  from  $a$  to  $b$ , including  $C_i$  itself. Let  $J_0$  (resp.  $J_{k+1}$ ) be the set consisting of only one cluster  $\{s, s\}$  (resp.  $\{t, t\}$ ).

Again, we first construct an atomic chain diagram  $G_i$ , for each  $i = 1, 2, \dots, k+1$ . The diagram  $G_i$  records all possible atomic chains on  $J_i$  and a conjugation relation from  $J_{i-1}$  to  $J_i$ . Since the points of clusters are arbitrarily given, we construct a new weighted diagram of  $G_i$ , denoted by  $GA_i$ . For an arc  $(x, y)$  of type (i),  $x, y \in J_i$ , its augmented atomic chain from  $x$  to  $y$  can be computed using the algorithm of Sect. 2.4.1. The weight of that arc in  $GA_i$  is assigned with the difference in length between the portion of  $U$  from  $x$  to  $y$  and the found chain or solution from  $x$  to  $y$ .

For an arc  $(u, v)$  of type (ii), there may be also some clusters of  $S$ , which do not belong to  $T$  and are contained in the convex region bounded by  $uv$  and the portion of  $U$  from  $u$  to  $v$ . Let us compute the convex hull of  $u, v$  and all points of those clusters. Again, it suffices to consider the clusters, whose point  $a$  or  $b$ , or both  $a$  and  $b$  are on the found hull. From Theorem 1, the augmented segment connecting  $u$  and  $v$  can be computed in  $O(n^2)$  time. The weight of arc  $(u, v)$  is then assigned with the difference in length between the portion of  $U$  and the found chain, from  $u$  to  $v$ . This gives the weights of all arcs of type (ii).

Finally, the weight of an arc of type (iii) in  $GW_i$  ( $1 \leq i \leq k+1$ ) represents an optimum solution from  $s$  to a point  $x \in J_{i-1}$ , in which the input is limited to the clusters contained in the region bounded by segment  $sx$  and the portion of  $U$  from  $s$  to  $x$ . As in the previous sections, it can be assigned analogously.

**Lemma 8.** *The weighted diagrams  $GA_1, GA_2, \dots, GA_{k+1}$  can be all constructed in  $O(n^4)$  time.*

**Proof.** Omitted in this extended abstract.  $\square$

**Theorem 2.** *The largest convex hull for a set of disjoint clusters of size two, measured by perimeter or area, can be computed in  $O(n^4)$  time.*

**Proof.** Omitted in this extended abstract.  $\square$

### 3 Extension

In this section, We modify our algorithms to present  $O(n^4)$  time solutions of the problems *MaxPCH* and *MaxACH*, for a set of disjoint clusters of constant size. We will focus on *MaxPCH*, as *MaxACH* can be dealt with analogously.

Let  $c$  be the bounded size of all clusters. Denote by  $S$  the set of disjoint clusters, and  $P$  the set of all given points. As in Sect. 2, we focus only on the clusters, which have at least two points on  $CH(P)$ . Let us fix a cluster, say,  $C$ . Let  $C_1, C_2 \dots, C_m$  be the sequence of clusters appearing on  $CH(P)$ , indexed by their *first* points on  $CH(P)$  clockwise, starting from some point of  $C$ .

Suppose first that all given points are in convex position, and the largest point of  $C_i$ ,  $1 \leq i \leq m - 1$ , is *not* adjacent to the smallest one of  $C_{i+1}$  on  $CH(P)$ . Denote by  $\mathcal{C}$  an optimum solution to this atomic version of *MaxPCH*. By considering every point of  $C$  as  $s$ , we can define at most  $c$  atomic problems  $MPCH(U, s, t)$ , where point  $t$  is an identical copy of  $s$ , and  $U$  is the convex hull (chain) of  $s$  and points of  $\{C_1 \cup \dots \cup C_m\}$ . The largest among at most  $c$  solutions then gives  $\mathcal{C}$ . Since every cluster is of constant size, as discussed in Sect. 2, the solution  $\mathcal{C}$  can be found in linear time.

Next, consider the situation in which all points of given clusters are in convex position. Again, by considering every point of a cluster as  $s$ , we can define at most  $c$  problems *MaxPCH*. (In the following, the cluster containing  $s$  needn't considered.) Let  $I$  be the set of clusters such that for any two consecutive clusters  $C$  and  $C'$  in  $I$ , the largest point of  $C$  is adjacent to the smallest one of  $C'$  on  $U$ . Suppose that  $I$  has  $k$  clusters, say,  $C'_1, C'_2, \dots, C'_k$ . For a cluster  $C'_i$ ,  $1 \leq i \leq k$ , denote by  $J_i$  the set of clusters (including  $C'_i$  itself), which have at least two points on the portion of  $U$  between the smallest and largest points of  $C'_i$ .

Assume that  $J_i$  ( $1 \leq i \leq k$ ) has  $m_i$  clusters  $C_{i,1}(=C'_i), C_{i,2}, \dots, C_{i,m_i}$ , which are ordered by their first points on chain  $U$ . For each cluster  $C_{i,h}$ ,  $1 \leq h \leq m_{i-1}$ , its points are further classified into two groups  $G_{i,h_1}$  and  $G_{i,h_2}$  such that all points  $u$  of  $G_{i,h_1}$  are before any point of  $C_{i,h+1}$  and all points  $v$  of  $G_{i,h_2}$  are after any point of  $C_{i,h+1}$  on  $U$ . Also, let  $G_{i,m_{i-1}}$  be the set consisting of only the smallest point of  $C_{i,m_i}$ , and  $G_{i,m_{i2}}$  the set consisting of all other points of  $C_{i,m_i}$ . Points  $u$  ( $\in G_{i,h_1}$ ) and  $v$  ( $\in G_{i,h_2}$ ),  $1 \leq h \leq m_i$ , then play the same role as points  $a$  and  $b$  respectively in the case of clusters of size two. Clearly, the number of atomic chains defined on  $J_i$  is no more than  $c|J_i|$ .

The diagram  $G_i$ ,  $1 \leq i \leq k + 1$ , can be constructed as follows. All points of  $J_i$  on chain  $U$  are added to the set  $V(G_i)$  as nodes. Note that the (unique) node  $u \in G_{i,m_{i-1}}$  has its copy  $u_{last2}$  and all nodes  $v \in G_{i,m_{i2}}$  have their copies  $v_{last2}$  in  $V(G_i)$ . Also,  $s$  is added to all sets  $V(G_i)$ , and  $t$  is added to  $V(G_{k+1})$ . Since  $u \in G_{i,h_1}$  and  $v \in G_{i,h_2}$ ) play the same role as  $a$  and  $b$  respectively in the case of clusters of size two, all arcs of three types are constructed analogously.

As in Sect. 2.3, the weighted diagrams  $GW_1, GW_2, \dots, GW_{k+1}$  can be constructed, too. Again, the following result holds.

**Lemma 9.** *For any starting point  $s$  of the fixed cluster, the optimum solution of  $MPCH(U, s, t)$  can be represented as a folded  $st$ -path in  $GW_{k+1}$ . Also, a folded  $st$ -path in  $GW_{k+1}$  corresponds to a feasible solution of  $MPCH(U, s, t)$ .*

**Theorem 3.** *Suppose that all given points are in convex position. The largest convex hull for a set of disjoint clusters of constant size, measured by perimeter or area, can be computed in  $O(n^2)$  time.*

**Proof.** For a starting point  $s$ , by an analogous to the proof of Theorem 1, the optimum solution of  $MPCH(U, s, t)$  can be found in  $O(n^2)$  time. Since the number of possible starting points  $s$  is no more than  $c$ , the theorem follows.  $\square$

For a set of clusters with arbitrarily given points, the similar argument can be given. Hence, the main result of this paper can be obtained.

**Theorem 4.** *The largest convex hulls for  $n$  disjoint clusters of bounded size, measured by perimeter or area, can be computed in  $O(n^4)$  time.*

**Remark B.** The problems  $MaxACH$  and  $MaxPCH$ , for a set of non-overlapping squares, have an optimum solution in which all vertices of the reported convex hull are the vertices of squares [7]. Our  $O(n^4)$  time algorithm works for both  $MaxACH$  and  $MaxPCH$  for a set of disjoint clusters of size four, improving upon the known time bounds  $O(n^5)$  [6] and  $O(n^{10})$  [7, Theorem 11], respectively.

## References

1. Abellanas, M., Hurtado, F., Ramos, P.A.: Structural tolerance and delaunay triangulation. *Inform. Process. Lett.* **71**, 221–227 (1999)
2. Corman, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: *An Introduction to Algorithms*, 3rd edn. The MIT Press, Cambridge (2009)
3. Díaz-Báñez, J.M., Korman, M., Pérez-Lantero, P., Pilz, A., Seara, C., Silveira, R.I.: New results on stabbing segments with a polygon. *Comput. Geom.* **48**, 14–29 (2015)
4. Dror, M., Efrat, A., Lubiw, A., Mitchell, J.S.B.: Touring a sequence of polygons. In: *Proceedings of the STOC'03*, pp. 473–482 (2003)
5. Guibas, L., Salesin, D., Stolfi, J.: Constructing strongly convex approximate hulls with inaccurate primitives. *Algorithmica* **9**, 534–560 (1993)
6. Ju, W., Luo, J., Zhu, B., Daescu, O.: Largest area convex hull of imprecise data based on axis-aligned squares. *J. Comb. Optim.* **26**, 832–859 (2013)
7. Löffler, M., van Kreveld, M.J.: Largest and Smallest convex hulls for imprecise points. *Algorithmica* **56**, 324–327 (2010)



# Two-Stage Submodular Maximization Under Knapsack and Matroid Constraints

Zhicheng Liu<sup>1</sup>, Jing Jin<sup>3</sup>, Donglei Du<sup>2</sup>, and Xiaoyan Zhang<sup>4</sup>(✉)

<sup>1</sup> Beijing Institute for Scientific and Engineering Computing, Beijing University of Technology, Beijing 100124, People's Republic of China

<sup>2</sup> Faculty of Management, University of New Brunswick, Fredericton, NB E3B 5A3, Canada  
ddu@unb.ca

<sup>3</sup> College of Taizhou, Nanjing Normal University, Taizhou 225300, China

<sup>4</sup> School of Mathematical Science and Institute of Mathematics, Nanjing Normal University, Nanjing 210023, China  
royxyzhang@gmail.com, zhangxiaoyan@njnu.edu.cn

**Abstract.** We consider the two-stage submodular maximization problem which has been studied extensively in machine learning. In this problem, we are given a ground set  $V$  of  $n$  articles and  $m$  categories, and the goal is to select a subset  $S \subseteq V$  of articles that best represents the different categories to maximize the total similarity  $F(S) = \sum_{j=1}^m \max_{T \in \mathcal{I}(S)} f_j(T)$ , where each  $f_j$  is a nonnegative monotone submodular functions measuring the similarity of each article in category  $j$ . We consider the case where  $S$  satisfies the knapsack constraint and  $\mathcal{I}(S)$  is a  $k$ -matroid constraint and present a  $\frac{1}{2(k+1)} (1 - e^{-(k+1)})$ -approximation algorithm for this problem. We also extend the  $k$ -matroid constraint to  $k$ -exchange system constraint and give corresponding approximation ratio.

**Keywords:** Submodular function · Knapsack constraint · Matroid

## 1 Introduction

The problem we are interested in is related to the Combinatorial Representation Problem (CRP). In CRP, we are given a ground set  $V$  of  $n$  articles and a set  $N$  of  $m$  categories, and nonnegative monotone submodular functions  $f_j : 2^V \rightarrow \mathbb{R}_{\geq 0}$  used to measure the similarity of each article in category  $j$ . The goal is to select a subset  $S \subseteq V$  of articles that best represents the different categories. This problem can be formulated as the following two-stage submodular maximization problem according to [1]:

$$\max_{S \in \mathcal{C}_1} F(S) = \max_{S \in \mathcal{C}_1} \frac{1}{m} \sum_{j \in V} \max_{T \in \mathcal{C}_2(S)} f_j(T), \quad (1.1)$$



where  $\mathcal{C}_1 \subseteq V$  and  $\mathcal{C}_2 \subseteq S$  are two constraint sets and  $f_j$  is a non-negative monotone submodular function. For example,  $\mathcal{C}_1$  may be a cardinality constraint and  $\mathcal{C}_2$  may be a matroid constraint [1, 14, 16]. To the best of our knowledge, there are no previous results on  $\mathcal{C}_1$  being a knapsack constraint which can be seen as an extension of the cardinality constraint and  $\mathcal{C}_2$  is a  $k$ -matroid constraint or a  $k$ -exchange system. Our problem is related to the Social Welfare Problem (SWP) which has been widely studied in economics. In SWP, they are given a set  $V$  of  $n$  items and a set  $B$  of  $m$  bidders. Each bidder  $i$  has a value function  $V_i : 2^V \rightarrow R_{\geq 0}$ . The goal is to allocate items to bidders by partitioning  $V$  into  $m$  sets  $V_1, \dots, V_m$  to maximize the objective  $\sum_{i=1}^m V_i(S_i)$ , where  $S_i$  is the set of items received by bidder  $i$ . While in Social Welfare Problem, the element can only be allocated to a bidder, in our problem, the element can be allocated to some categories.

This problem involves two-stage optimization: (1) the second stage (inner problem) maximizes a monotonic submodular function subject to some constraints, resulting in the optimal objective value being a function of  $S$ ; and (2) the first stage maximizes this resulted objective subject to some other constraints.

Evidently, the difficulty in solving this two-stage submodular maximization problem (1.1) depends on both the types of objective functions  $f_j$  and the constraints  $\mathcal{C}_1$  and  $\mathcal{C}_2$ .

In this paper, we assume that  $\mathcal{C}_1$  is a knapsack constraint and  $\mathcal{C}_2 = \mathcal{I}(S)$  is the family of the common independent sets of  $k$  matroid  $\mathcal{M}_j = (S, \mathcal{I}_j(S))$  over the same ground set  $S \subseteq V$ ; namely  $\mathcal{I}(S) = \cap_{j=1}^k \mathcal{I}_j(S)$ . Our contribution is to present a  $\frac{1}{2^{(k+1)}} (1 - e^{-(k+1)})$ -approximation algorithm for this problem. Considering the  $k$ -exchange system constraint, we also give some similar results.

The rest of the paper is organized as follows. In Sect. 3 we introduce some definitions and properties of submodular function and matroid. In Sect. 4, 5, we present the algorithms and analysis of Problem (1.1). Finally, we offer concluding remarks in Sect. 6.

## 2 Related Work

The combinatorial representation problem can be seen as a two-stage submodular maximization problem [1, 14, 16]. The authors in [1, 14, 16] consider the problem of submodular maximization where the objective function  $f$  satisfies the uniform distribution. In [1], they gave two algorithms based on the techniques of continuous optimization and local search. However, both run times are very expensive. Recently, the authors in [16] use a simple greedy algorithm to improve the approximation ratio from  $\frac{1}{2} (1 - e^{-1})$  to  $\frac{1}{2} (1 - e^{-2})$ . The authors in [14] develop the first streaming and distributed algorithms for this problem. In addition, authors in [18] extend the matroid constraint to  $k$ -matroids constraint.

The submodular social welfare problem has been widely studied in both offline and online settings. For the offline version, [4, 8] presents a  $(1 - e^{-1})$ -approximation algorithm via the multilinear extension and [10] proves that this

ratio is the best possible unless  $P=NP$ . For the online version, where the items arrive in a random order, [6, 7] give a  $\ell/(2\ell - 1)$ -approximation algorithm, while [11] offers a 0.5052-competitive ratio by a simple greedy algorithm. The latter algorithm can be seen as the first  $(1/2 + \Omega(1))$ -competitive algorithm for this online setting. Recently, [2] delivers a simpler analysis and improves the competitive ratio to 0.5096. Another online setting is to assume the items arrive in an adversarial way [5, 9].

### 3 Preliminaries

Given a ground set  $V = \{1, \dots, n\}$  and a family of subsets  $\mathcal{I}$  of  $V$ , a matroid  $\mathcal{M} = (V, \mathcal{I})$  satisfies the following properties

- (1) If  $A \subseteq B \in \mathcal{I}$ , then  $A \in \mathcal{I}$ ;
- (2) If  $A, B \in \mathcal{I}$  and  $|A| \leq |B|$ , then there exists an element  $u \in B \setminus A$  for which  $A + u \in \mathcal{I}$ .

A partition matroid has an independent set  $\mathcal{I} = \{I : \forall j; |I \cap P_j| \leq r_j\}$ , where  $P = P_1 \cup P_2 \cup \dots \cup P_m$  is a disjoint union, and  $r_1, r_2, \dots, r_m \in \mathbb{Z}_+$ .

We will need the following matroid property from [13] later.

**Property 1.** *Let  $\mathcal{M}_j = (V, \mathcal{I}_j)$  be a matroid for every  $j \in \{1, \dots, k\}$ . For any two independent sets  $A, B \in \mathcal{I}_j$ , there exists a mapping  $\pi_j : B \setminus A \rightarrow A \setminus B \cup \{\emptyset\}$  such that*

- (1)  $(A \setminus \pi_j(b)) \cup b \in \mathcal{I}_j$  for all  $b \in B \setminus A$ ;
- (2)  $|\pi_j^{-1}(a)| \leq 1$  for all  $a \in A \setminus B$ ;
- (3) let  $A_b = \{\pi_1(b), \dots, \pi_k(b)\}$ , then  $(A \setminus A_b) \cup b \in \cap_{j=1}^k \mathcal{I}_j$  for all  $b \in B \setminus A$ .

Next, we introduce two equivalent definitions and properties of submodular function.

**Definition 1.** *A function  $f: 2^V \rightarrow \mathbb{R}_+$  is submodular if for every  $X, Y \subseteq V$ ,*

$$f(X) + f(Y) \geq f(X \cup Y) + f(X \cap Y).$$

**Definition 2.** *A function  $f: 2^V \rightarrow \mathbb{R}_+$  is submodular if for every  $X \subseteq Y \subseteq V, a \in V \setminus Y$ ,*

$$f(X \cup \{a\}) - f(X) \geq f(Y \cup \{a\}) - f(Y).$$

**Property 2.** *For any submodular function  $f: 2^V \rightarrow \mathbb{R}_+$  and  $X, Y \subseteq V$ , we have*

$$\sum_{u \in X} f(Y \cup \{u\}) - f(Y) \geq f(X \cup Y) - f(Y).$$

The next property is from [17].

**Property 3.** Consider a monotone submodular function  $f: 2^V \rightarrow \mathbb{R}_+$ . Let  $X, Y \subseteq V$ , and  $\{T_i\}_{i=1}^\ell$  be a collection of subsets of  $Y \setminus X$  such that each element of  $Y \setminus X$  appears in at most  $k$  of the subsets. Then

$$\sum_{i=1}^{\ell} [f(Y) - f(Y \setminus T_i)] \leq k[f(Y) - f(Y \cap X)].$$

We recall that a pair  $(S, \mathcal{I})$  is called an independence system, if for  $I_1 \subseteq I_2 \subseteq S$  and  $I_2 \in \mathcal{I}$ , then  $I_1 \in \mathcal{I}$ . A more generalized  $P$ -exchange system can be formally redefined as follows.

**Definition 3.** The independence system  $(S, \mathcal{I})$  is called  $k$ -exchange system, if for  $\forall X, Y \in \mathcal{I}$ ,  $A = \{A_b \subseteq X \setminus Y : b \in Y \setminus X\}$  satisfies the following three conditions:

- (1) for  $\forall b \in Y \setminus X$ ,  $|A_b| \leq k$ ;
- (2) for  $\forall a \in X \setminus Y$ , it is contained in at most  $k$  sets in  $A$ ,
- (3) for  $Y' \subseteq Y$ ,  $X - \cup_{b \in Y'} A_b + Y' \in \mathcal{I}$ .

In addition, for any  $A, B \in \mathcal{I}$ , we say  $B$  is an extension of  $A$  if  $A \subseteq B$ . There is a more generalized  $P$ -extendible system restated as below.

**Definition 4.** An independence system  $(S, \mathcal{I})$  is a  $P$ -extendible system if for every independent set  $A \in \mathcal{I}$ , any extension  $B$  of  $A$  and an element  $x \notin A$  obeying  $A + x \in \mathcal{I}$ , there must exist a subset  $Y \subseteq B \setminus A$  with  $|Y| \leq P$  such that  $B + x - Y \in \mathcal{I}$ . Specially, if  $B = A$  or  $x \in B$ , then we can take  $Y = \emptyset$ .

## 4 Two-Stage Submodular Maximization Subject to Knapsack and $K$ -Matroid Constraints

We consider Problem (1.1) by offering an approximation algorithm along with its analysis in Sects. 5.1 and 5.2, respectively.

### 4.1 The Algorithm

Denote the gain of adding element  $x$  to the set  $A$  as follows:

$$\Delta_j^f(x, A) = f_j(\{x\} \cup A) - f_j(A).$$

Denote the gain of removing a set  $Y \subseteq A$  and replacing it with  $x$  as follows:

$$\nabla_j^f(x, Y, A) = f_j(\{x\} \cup A \setminus Y) - f_j(A).$$

The set of elements in  $A$  can replace  $x$  which will not violate the  $k$ -matroid constraint, is defined as

$$\mathcal{I}(x, A) = \{Y \subseteq A : A \cup \{x\} \setminus Y \in \mathcal{I}\}$$

. Define the replacement gain of  $x$  as follow:

$$\nabla_j^f(x, A) = \begin{cases} \Delta_j^f(x, A), & \text{if } A \cup \{x\} \in \mathcal{I}, \\ \max\{0, \max_{Y \in \mathcal{I}(x, A)} \nabla_j^f(x, Y, A)\}, & \text{otherwise.} \end{cases}$$

Let  $\text{Rep}_j^f(x, A)$  be the set that is replaced by  $x$ :

$$\text{Rep}_j^f(x, A) = \begin{cases} \emptyset, & \text{if } A \cup \{x\} \in \mathcal{I}, \\ \arg \max_{Y \in \mathcal{I}(x, A)} \nabla_j^f(x, Y, A), & \text{otherwise.} \end{cases}$$

The algorithm starts with an empty set  $S = \emptyset$ , and chooses an element with the largest ratio of marginal gain over cost in every round. Property 1 guarantees the correctness of the our algorithm.

---

### Algorithm 1 Replacement Modified Greedy

---

```

1:  $U \leftarrow V, S^F \leftarrow \emptyset, S \leftarrow \emptyset, T_j^F \leftarrow \emptyset, T_j \leftarrow \emptyset (\forall 1 \leq j \leq m)$ 
2: while  $U \neq \emptyset$  do
3:    $k \leftarrow \arg \max_{e \in U} \frac{\sum_{j=1}^m \nabla_j^f(e, T_j)}{c_e}$ 
4:    $U \leftarrow U \setminus \{k\}$ 
5:   if  $\sum_{u \in S} c_u + c_k \leq B$  then
6:      $S \leftarrow S \cup \{k\}$ 
7:     for all  $1 \leq j \leq m$  do
8:       if  $\nabla_j^f(k, T_j) > 0$  then
9:          $T_j \leftarrow T_j \cup \{k\} \setminus \text{Rep}_j^f(k, T_j)$ 
10:      end if
11:    end for
12:  end if
13: end while
14:  $u^* \leftarrow \arg \max_{u \in V, c_u \leq B} \sum_{j=1}^m f_j(u^*)$ 
15: if  $\sum_{j=1}^m f_j(T_j) > \sum_{j=1}^m f_j(u^*)$  then
16:    $S^F \leftarrow S, T_1^F \leftarrow T_1, T_2^F \leftarrow T_2, \dots, T_m^F \leftarrow T_m$ 
17: else
18:    $S^F \leftarrow u^*, T_1^F \leftarrow u^*, T_2^F \leftarrow u^*, \dots, T_m^F \leftarrow u^*$ 
19: end if

```

---

## 4.2 The Analysis

Define  $S^*$  as the optimal solution of problem (1.1):

$$S^* = \arg \max_{c(S) \leq B} \sum_{j=1}^m \frac{1}{m} \max_{T \in \mathcal{I}(S)} f_j(T).$$

Denote  $S_j^*$  as the optimal solution of  $f_j$ :

$$S_j^* = \arg \max_{T \in I(S^*)} f_j(T).$$

Based on Algorithm 1, we introduce the following notations.

- $S$  is the solution obtained by the greedy heuristic;
- $v_i$  is the  $i$ th unit added to  $S$  ( $i = 1, \dots, |S|$ );
- $S_i$  is the set of function  $F(S)$  obtained by greedy algorithm after adding  $v_i$  (i.e.,  $S_i = \cup_{k=1}^i \{v_k\}$ , for  $i = 1, \dots, |S|$ , with  $S_0 = \emptyset$ ,  $S_{|S|} = S$ ); and
- $T_j^i$  is the set which is chosen by  $f_j(T)$  after adding  $v_i$  to the set  $S$ .

We first establish the following two lemmas to bound the increment in each iteration.

**Lemma 1.** For  $i = 1, 2, \dots, |S| + 1$ , we have

$$\sum_{e \in S^*} \sum_{j=1}^m \nabla_j^f(e, T_j^{i-1}) \leq \frac{B}{c_{v_i}} \sum_{j=1}^m \nabla_j^f(v_i, T_j^{i-1}).$$

*Proof.* From Line 3 of Algorithm 1, we have

$$\frac{\sum_{j=1}^m \nabla_j^f(e, T_j^{i-1})}{c_e} \leq \frac{\sum_{j=1}^m \nabla_j^f(v_i, T_j^{i-1})}{c_{v_i}}, \forall e \in S^*.$$

Thus,

$$\sum_{e \in S^*} \sum_{j=1}^m \nabla_j^f(e, T_j^{i-1}) \leq \sum_{j=1}^m \nabla_j^f(v_i, T_j^{i-1}) \frac{\sum_{e \in S^*} c_e}{c_{v_i}} \leq \frac{B}{c_{v_i}} \sum_{j=1}^m \nabla_j^f(v_i, T_j^{i-1}).$$

Where the last inequality follows from  $\sum_{e \in S^*} c_e \leq B$ . ■

**Lemma 2.** For  $i = 1, 2, \dots, |S| + 1$ , we have

$$\frac{B}{c_{v_i}} \sum_{j=1}^m \nabla_j^f(v_i, T_j^{i-1}) \geq \sum_{j=1}^m (f_j(S_j^*) - (k+1)f_j(T_j^{i-1})).$$

*Proof.* The proof is in the appendix. ■

For convenience, we denote  $X_{i-1} := \sum_{j=1}^m f_j(T_j^{i-1})$  and  $X^* := \sum_{j=1}^m f_j(S_j^*)$ . So the left item  $\sum_{j=1}^m \nabla_j^f(v_i, T_j^{i-1})$  in the inequality of Lemma 2 can be written as  $\frac{B}{c_{v_i}}(X_i - X_{i-1})$ , and the right item  $\sum_{j=1}^m (f_j(S_j^*) - (k+1)f_j(T_j^{i-1}))$  can be written as  $X^* - (k+1)X_{i-1}$ , and Lemma 2 can be written as

$$X_i - X_{i-1} \geq \frac{c_{v_i}}{B}(X^* - (k+1)X_{i-1}). \quad (4.1)$$

**Lemma 3.**  $\forall i = 1, 2, \dots, |S| + 1$ , if we assume  $X^* \geq (k + 1)X_i$ , then we have

$$(k + 1)c_{v_i} \leq B, \quad \forall i = 1, 2, \dots, |S| + 1$$

*Proof.* The proof is in the appendix. ■

According to the Lemma 3, we have the following result.

**Lemma 4.**  $\forall i = 1, 2, \dots, |S| + 1$ , if we assume  $X^* \geq (k + 1)X_i$ , then we have

$$X_{|S|+1} \geq \frac{1}{k + 1} \left(1 - e^{-(k+1)}\right) X^*,$$

*Proof.* The proof is in the appendix. ■

**Theorem 1.** Algorithm 1 returns a set  $S_F$  such that

$$F(S_F) \geq \frac{1}{2(k + 1)} \left(1 - e^{-(k+1)}\right) F(S^*).$$

*Proof.* We consider two cases.

Case 1: There exists a  $t$  such that  $X^* \leq (k + 1)X_t$ . Then

$$\begin{aligned} F(S_F) &\geq F(X_t) \geq \frac{1}{k + 1} F(S^*) \\ &\geq \frac{1}{2(k + 1)} \left(1 - e^{-(k+1)}\right) F(S^*). \end{aligned}$$

where the first inequality follows from  $X_t \subseteq S_F$ .

Case 2:  $\forall i = 1, 2, \dots, |S| + 1$ , we have  $X^* \geq (k + 1)X_i$ .

According to the Lemma 4, we have

$$X_{|S|+1} \geq \frac{1}{k + 1} \left(1 - e^{-(k+1)}\right) X^*,$$

From Algorithm 1, we also have

$$\begin{aligned} X_{|S|+1} - X_{|S|} &\leq \sum_{j=1}^m \left( f_j(T_j^{|S|} \cup \{v_{|S|+1}\}) - f_j(T_j^{|S|}) \right) \\ &\leq \sum_{j=1}^m \left( f_j(\{v_{|S|+1}\}) - f_j(\emptyset) \right) \\ &= \sum_{j=1}^m \left( f_j(\{v_{|S|+1}\}) \right) \\ &\leq \sum_{j=1}^m f_j(\{u^*\}), \end{aligned}$$

where the first inequality is due to Line 9 of Algorithm 1 and the second inequality follows from the submodularity of  $f_j$ .

Hence,

$$\sum_{j=1}^m f_j(\{u^*\}) + X_{|S|} \geq X_{|S|+1} \geq \frac{1}{k+1} \left(1 - e^{-(k+1)}\right) X^*,$$

implying that

$$\max \left\{ \sum_{j=1}^m f_j(\{u^*\}), X_{|S|} \right\} \geq \frac{1}{2(k+1)} \left(1 - e^{-(k+1)}\right) X^*.$$

■

## 5 Two-Stage Submodular Maximization Subject to Knapsack and $k$ -exchange System Constraints

In this section, we consider Problem (1.1) under knapsack and  $k$ -exchange system constraints by offering an approximation algorithm along with its analysis in Sects. 5.1 and 5.2, respectively.

### 5.1 The Algorithm

The set of elements in  $A$  can replace  $x$  which will not violate the  $k$ -exchange system constraint, is defined as

$$\tilde{\mathcal{I}}(x, A) = \{Y \subseteq A : A \cup \{x\} \setminus Y \in \mathcal{I}\}.$$

Define the replacement gain of  $x$  as follow:

$$\tilde{\nabla}_j^f(x, A) = \begin{cases} \Delta_j^f(x, A), & \text{if } A \cup \{x\} \in \mathcal{I}, \\ \max\{0, \max_{Y \in \tilde{\mathcal{I}}(x, A)} \nabla_j^f(x, Y, A)\}, & \text{otherwise.} \end{cases}$$

Let  $\widetilde{\text{Rep}}_j^f(x, A)$  be the set that is replaced by  $x$ :

$$\widetilde{\text{Rep}}_j^f(x, A) = \begin{cases} \emptyset, & \text{if } A \cup \{x\} \in \mathcal{I}, \\ \arg \max_{Y \in \tilde{\mathcal{I}}(x, A)} \nabla_j^f(x, Y, A), & \text{otherwise.} \end{cases}$$

The algorithm starts with an empty set  $S = \emptyset$ , and chooses an element with the largest ratio of marginal gain over cost in every round. Property 1 guarantees the correctness of the our algorithm.

**Algorithm 2** Replacement Modified Greedy

---

```

1:  $U \leftarrow V, S^F \leftarrow \emptyset, S \leftarrow \emptyset, T_j^F \leftarrow \emptyset, T_j \leftarrow \emptyset (\forall 1 \leq j \leq m)$ 
2: while  $U \neq \emptyset$  do
3:    $k \leftarrow \arg \max_{e \in U} \frac{\sum_{j=1}^m \tilde{\nabla}_j^f(e, T_j)}{c_e}$ 
4:    $U \leftarrow U \setminus \{k\}$ 
5:   if  $\sum_{u \in S} c_u + c_k \leq B$  then
6:      $S \leftarrow S \cup \{k\}$ 
7:     for all  $1 \leq j \leq m$  do
8:       if  $\tilde{\nabla}_j^f(k, T_j) > 0$  then
9:          $T_j \leftarrow T_j \cup \{k\} \setminus \widetilde{\text{Rep}}_j^f(k, T_j)$ 
10:        end if
11:      end for
12:    end if
13:  end while
14:  $u^* \leftarrow \arg \max_{u \in V, c_u \leq B} \sum_{j=1}^m f_j(u^*)$ 
15: if  $\sum_{j=1}^m f_j(T_j) > \sum_{j=1}^m f_j(u^*)$  then
16:    $S^F \leftarrow S, T_1^F \leftarrow T_1, T_2^F \leftarrow T_2, \dots, T_m^F \leftarrow T_m$ 
17: else
18:    $S^F \leftarrow u^*, T_1^F \leftarrow u^*, T_2^F \leftarrow u^*, \dots, T_m^F \leftarrow u^*$ 
19: end if

```

---

**5.2 The Analysis**

We first establish the following two lemmas to bound the increment in each iteration.

**Lemma 5.** For  $i = 1, 2, \dots, |S| + 1$ , we have

$$\sum_{e \in S^*} \sum_{j=1}^m \tilde{\nabla}_j^f(e, T_j^{i-1}) \leq \frac{B}{c_{v_i}} \sum_{j=1}^m \tilde{\nabla}_j^f(v_i, T_j^{i-1}).$$

*Proof.* From Line 3 of Algorithm 1, we have

$$\frac{\sum_{j=1}^m \tilde{\nabla}_j^f(e, T_j^{i-1})}{c_e} \leq \frac{\sum_{j=1}^m \tilde{\nabla}_j^f(v_i, T_j^{i-1})}{c_{v_i}}, \forall e \in S^*.$$

Thus,

$$\sum_{e \in S^*} \sum_{j=1}^m \tilde{\nabla}_j^f(e, T_j^{i-1}) \leq \sum_{j=1}^m \tilde{\nabla}_j^f(v_i, T_j^{i-1}) \frac{\sum_{e \in S^*} c_e}{c_{v_i}} \leq \frac{B}{c_{v_i}} \sum_{j=1}^m \tilde{\nabla}_j^f(v_i, T_j^{i-1}).$$

Where the last inequality follows from  $\sum_{e \in S^*} c_e \leq B$ . ■

**Lemma 6.** For  $i = 1, 2, \dots, |S| + 1$ , we have

$$\frac{B}{c_{v_i}} \sum_{j=1}^m \tilde{\nabla}_j^f(v_i, T_j^{i-1}) \geq \sum_{j=1}^m (f_j(S_j^*) - (k+1)f_j(T_j^{i-1})).$$



*Proof.* The proof is in the appendix. ■

For convenience, the Lemma 2 can be written as

$$X_i - X_{i-1} \geq \frac{c_{v_i}}{B} (X^* - (k+1)X_{i-1}). \quad (5.1)$$

According to the above Lemmas, we have the following result.

**Theorem 2.** *Algorithm 2 returns a set  $S_F$  such that*

$$F(S_F) \geq \frac{1}{2(k+1)} \left(1 - e^{-(k+1)}\right) F(S^*).$$

*Proof.* The proof is in the appendix. ■

## 6 Conclusion

In this paper, we consider a two-stage submodular maximization problem and give the first constant approximation algorithm under the knapsack constraint and  $k$ -matroid constraint. An interesting future problem is to design an efficient algorithm under the online and distributed setting.

**Acknowledgment.** The research is partially supported by NSFC (Nos. 12131003, 11871280, 12271259, 11971349), the Natural Sciences and Engineering Research Council of Canada (NSERC) Grant 06446, Natural Science Foundation of Jiangsu Province (No. BK20200267), Qinglan Project of Education Department of Jiangsu Province and Jiangsu Province Higher Education Foundation (No.20KJB110022).

## A Appendix

The first is the proof of Lemma 2.

*Proof.* Lemma 1 implies that

$$\sum_{e \in S^*} \sum_{j=1}^m \nabla_j^f(e, T_j^{i-1}) \leq \frac{B}{c_{v_i}} \sum_{j=1}^m \nabla_j^f(v_i, T_j^{i-1}).$$

From Property 1, there exist mappings  $\pi_t : S_j^* \setminus T_j^{i-1} \rightarrow T_j^{i-1} \setminus S_j^* \cup \{\emptyset\}$  ( $t \in \{1, 2, \dots, k\}$ ) such that  $(T_j^{i-1} \setminus A_e) \cup \{e\} \in \cap_{t=1}^k \mathcal{I}_t$ , where  $A_e = \{\pi_1(e), \dots, \pi_k(e)\}$ . Therefore,

$$\begin{aligned}
& \sum_{e \in S^*} \sum_{j=1}^m \nabla_j^f(e, T_j^{i-1}) \\
&= \sum_{j=1}^m \sum_{e \in S^*} \nabla_j^f(e, T_j^{i-1}) \\
&\geq \sum_{j=1}^m \sum_{e \in S_j^* \setminus T_j^{i-1}} \nabla_j^f(e, T_j^{i-1}) \\
&\geq \sum_{j=1}^m \sum_{e \in S_j^* \setminus T_j^{i-1}} \left( f_j(\{e\} \cup T_j^{i-1} \setminus A_e) - f_j(T_j^{i-1}) \right) \\
&= \sum_{j=1}^m \sum_{e \in S_j^* \setminus T_j^{i-1}} \left( f_j(\{e\} \cup T_j^{i-1} \setminus A_e) - f_j(\{e\} \cup T_j^{i-1}) + f_j(\{e\} \cup T_j^{i-1}) - f_j(T_j^{i-1}) \right) \\
&= \sum_{j=1}^m \sum_{e \in S_j^* \setminus T_j^{i-1}} \left[ \Delta_j^f(e, T_j^{i-1}) - \left( f_j(\{e\} \cup T_j^{i-1}) - f_j(\{e\} \cup T_j^{i-1} \setminus A_e) \right) \right]
\end{aligned}$$

where the first inequality follows because  $\nabla_j^f(e, T_j^{i-1}) \geq 0$  and  $S_j^* \setminus T_j^{i-1} \subseteq S^*$ , and the second is due to Property 1 and the definition of  $\nabla_j^f(e, T_j^{i-1})$ .

Denote  $A_e^r = \{\pi_1(e), \dots, \pi_r(e)\}$ , for  $r = 1, \dots, k$  and  $A_e^0 = \emptyset$ ,  $A_e^k = A_e$ . We have

$$\begin{aligned}
& f_j(\{e\} \cup T_j^{i-1}) - f_j(\{e\} \cup T_j^{i-1} \setminus A_e) \\
&= f_j(\{e\} \cup T_j^{i-1}) - f_j(\{e\} \cup T_j^{i-1} \setminus \{\pi_1(e), \dots, \pi_k(e)\}) \\
&= \sum_{r=1}^k \left( f_j(\{e\} \cup T_j^{i-1} \setminus A_e^{r-1}) - f_j(\{e\} \cup T_j^{i-1} \setminus A_e^r) \right) \\
&\leq \sum_{r=1}^k \left( f_j(T_j^{i-1} \setminus A_e^{r-1}) - f_j(T_j^{i-1} \setminus A_e^r) \right) \\
&= f_j(T_j^{i-1}) - f_j(T_j^{i-1} \setminus A_e) \\
&= \Delta_j^f(A_e, T_j^{i-1} \setminus A_e),
\end{aligned}$$

where the inequality is from the submodularity of  $f_j$ .

So we have

$$\sum_{e \in S^*} \sum_{j=1}^m \nabla_j^f(e, T_j^{i-1}) \geq \sum_{j=1}^m \sum_{e \in S_j^* \setminus T_j^{i-1}} \left( \Delta_j^f(e, T_j^{i-1}) - \Delta_j^f(A_e, T_j^{i-1} \setminus A_e) \right).$$

Property 2 implies that

$$\sum_{e \in S_j^* \setminus T_j^{i-1}} \Delta_j^f(e, T_j^{i-1}) = \sum_{e \in S_j^* \setminus T_j^{i-1}} \left( f_j(e \cup T_j^{i-1}) - f_j(T_j^{i-1}) \right) \geq f_j(S_j^* \cup T_j^{i-1}) - f_j(T_j^{i-1}).$$

Property 3 implies that

$$\begin{aligned} \sum_{e \in S_j^* \setminus T_j^{i-1}} \Delta_j^f(A_e, T_j^{i-1} \setminus A_e) &= \sum_{e \in S_j^* \setminus T_j^{i-1}} (f_j(T_j^{i-1}) - f_j(T_j^{i-1} \setminus A_e)) \\ &\leq k (f_j(T_j^{i-1}) - f_j(T_j^{i-1} \cap S_j^*)) \\ &\leq k f_j(T_j^{i-1}). \end{aligned}$$

Together, we have

$$\begin{aligned} \sum_{e \in S^*} \sum_{j=1}^m \nabla_j^f(e, T_j^{i-1}) &\geq \sum_{j=1}^m (f_j(S_j^* \cup T_j^{i-1}) - (k+1)f_j(T_j^{i-1})) \\ &\geq \sum_{j=1}^m (f_j(S_j^*) - (k+1)f_j(T_j^{i-1})). \end{aligned}$$

■

The next is the proof of Lemma 3.

*Proof.* Suppose for contradiction that there exists  $j \leq |S| + 1$  such that  $(k+1)c_{v_i} > B$ , then

$$\begin{aligned} X_j &\geq \frac{c_{v_i}}{B} (X^* - (k+1)X_{j-1}) + X_{j-1} \\ &> \frac{1}{k+1} (X^* - (k+1)X_{j-1}) + X_{j-1} \\ &= \frac{1}{k+1} X^*, \end{aligned}$$

which contradicts the assumption. ■

The next is the proof of Lemma 4.

*Proof.* Rearranging the inequality (5.1), we obtain

$$\frac{\frac{1}{k+1}X^* - X_i}{\frac{1}{k+1}X^* - X_{i-1}} \leq 1 - \frac{(k+1)c_{v_i}}{B}.$$

Therefore,

$$\begin{aligned} \frac{1}{k+1}X^* - X_{|S|+1} &\leq \prod_{i=1}^{|S|+1} \left( 1 - \frac{(k+1)c_{v_i}}{B} \right) \frac{1}{k+1}X^* \\ &\leq \prod_{i=1}^{|S|+1} e^{-\frac{(k+1)c_{v_i}}{B}} \frac{1}{k+1}X^* \\ &= e^{-\frac{\sum_{i=1}^{|S|+1} (k+1)c_{v_i}}{B}} \frac{1}{k+1}X^* \\ &\leq e^{-\frac{(k+1)B}{B}} \frac{1}{k+1}X^* \\ &= e^{-(k+1)} \frac{1}{k+1}X^*, \end{aligned}$$

which is equivalent to

$$X_{|S|+1} \geq \frac{1}{k+1} \left(1 - e^{-(k+1)}\right) X^*,$$

where the second inequality above holds because  $1 - x \leq e^{-x}$  and the third inequality is due to  $\sum_{i=1}^{|S|+1} c_{v_i} > B$ . ■

The first is the proof of Lemma 5.

*Proof.* Lemma 1 implies that

$$\sum_{e \in S^*} \sum_{j=1}^m \tilde{\nabla}_j^f(e, T_j^{i-1}) \leq \frac{B}{c_{v_i}} \sum_{j=1}^m \tilde{\nabla}_j^f(v_i, T_j^{i-1}).$$

From Definition 3, we have

$$\begin{aligned} & \sum_{e \in S^*} \sum_{j=1}^m \tilde{\nabla}_j^f(e, T_j^{i-1}) \\ &= \sum_{j=1}^m \sum_{e \in S^*} \tilde{\nabla}_j^f(e, T_j^{i-1}) \\ &\geq \sum_{j=1}^m \sum_{e \in S_j^* \setminus T_j^{i-1}} \tilde{\nabla}_j^f(e, T_j^{i-1}) \\ &\geq \sum_{j=1}^m \sum_{e \in S_j^* \setminus T_j^{i-1}} \left( f_j(\{e\} \cup T_j^{i-1} \setminus A_e) - f_j(T_j^{i-1}) \right) \\ &= \sum_{j=1}^m \sum_{e \in S_j^* \setminus T_j^{i-1}} \left( f_j(\{e\} \cup T_j^{i-1} \setminus A_e) - f_j(\{e\} \cup T_j^{i-1}) + f_j(\{e\} \cup T_j^{i-1}) - f_j(T_j^{i-1}) \right) \\ &= \sum_{j=1}^m \sum_{e \in S_j^* \setminus T_j^{i-1}} \left[ \Delta_j^f(e, T_j^{i-1}) - \left( f_j(\{e\} \cup T_j^{i-1}) - f_j(\{e\} \cup T_j^{i-1} \setminus A_e) \right) \right] \end{aligned}$$

where the first inequality follows because  $\tilde{\nabla}_j^f(e, T_j^{i-1}) \geq 0$  and  $S_j^* \setminus T_j^{i-1} \subseteq S^*$ , and the second is due to Definition 3 and the definition of  $\tilde{\nabla}_j^f(e, T_j^{i-1})$ .

Denote  $A_e^r = \{\pi_1(e), \dots, \pi_r(e)\}$ , for  $r = 1, \dots, k$  and  $A_e^0 = \emptyset$ ,  $A_e^k = A_e$ . We have

$$\begin{aligned} & f_j(\{e\} \cup T_j^{i-1}) - f_j(\{e\} \cup T_j^{i-1} \setminus A_e) \\ &= f_j(\{e\} \cup T_j^{i-1}) - f_j(\{e\} \cup T_j^{i-1} \setminus \{\pi_1(e), \dots, \pi_k(e)\}) \\ &= \sum_{r=1}^k \left( f_j(\{e\} \cup T_j^{i-1} \setminus A_e^{r-1}) - f_j(\{e\} \cup T_j^{i-1} \setminus A_e^r) \right) \\ &\leq \sum_{r=1}^k \left( f_j(T_j^{i-1} \setminus A_e^{r-1}) - f_j(T_j^{i-1} \setminus A_e^r) \right) \\ &= f_j(T_j^{i-1}) - f_j(T_j^{i-1} \setminus A_e) \\ &= \Delta_j^f(A_e, T_j^{i-1} \setminus A_e), \end{aligned}$$

where the inequality is from the submodularity of  $f_j$ .

So we have

$$\sum_{e \in S^*} \sum_{j=1}^m \nabla_j^f(e, T_j^{i-1}) \geq \sum_{j=1}^m \sum_{e \in S_j^* \setminus T_j^{i-1}} \left( \Delta_j^f(e, T_j^{i-1}) - \Delta_j^f(A_e, T_j^{i-1} \setminus A_e) \right).$$

Property 2 implies that

$$\sum_{e \in S_j^* \setminus T_j^{i-1}} \Delta_j^f(e, T_j^{i-1}) = \sum_{e \in S_j^* \setminus T_j^{i-1}} \left( f_j(e \cup T_j^{i-1}) - f_j(T_j^{i-1}) \right) \geq f_j(S_j^* \cup T_j^{i-1}) - f_j(T_j^{i-1}).$$

Property 3 implies that

$$\begin{aligned} \sum_{e \in S_j^* \setminus T_j^{i-1}} \Delta_j^f(A_e, T_j^{i-1} \setminus A_e) &= \sum_{e \in S_j^* \setminus T_j^{i-1}} \left( f_j(T_j^{i-1}) - f_j(T_j^{i-1} \setminus A_e) \right) \\ &\leq k \left( f_j(T_j^{i-1}) - f_j(T_j^{i-1} \cap S_j^*) \right) \\ &\leq k f_j(T_j^{i-1}). \end{aligned}$$

Together, we have

$$\begin{aligned} \sum_{e \in S^*} \sum_{j=1}^m \tilde{\nabla}_j^f(e, T_j^{i-1}) &\geq \sum_{j=1}^m \left( f_j(S_j^* \cup T_j^{i-1}) - (k+1) f_j(T_j^{i-1}) \right) \\ &\geq \sum_{j=1}^m \left( f_j(S_j^*) - (k+1) f_j(T_j^{i-1}) \right). \end{aligned}$$

■

The next is the proof of Theorem 2.

*Proof.* The proof is the same as Theorem 1. ■




## References

1. Balkanski, E., Mirzasoleiman, B., Krause, A., Singer, Y.: Learning sparse combinatorial representations via two-stage submodular maximization. In: ICML, pp. 2207–2216 (2016)
2. Buchbinder, N., Feldman, M., Filmus, Y., Garg, M.: Online submodular maximization: beating 1/2 made simple. Math. Program. **183**(1), 149–169 (2020). <https://doi.org/10.1007/s10107-019-01459-z>
3. Buchbinder, N., Feldman, M., Naor, J., Schwartz, R.: A tight linear time (1/2)-approximation for unconstrained submodular maximization. SIAM J. Comput. **44**(5), 1384–1402 (2015)
4. Calinescu, G., Chekuri, C., Pal, M., Vondrák, J.: Maximizing a monotone submodular function subject to a matroid constraint. SIAM J. Comput. **40**(6), 1740–1766 (2011)

5. Devanur, N.R., Hayes, T.P.: The adwords problem: online keyword matching with budgeted bidders under random permutations. In: EC, pp. 71–78 (2009)
6. Dobzinski, S., Nisan, N., Schapira, M.: Approximation algorithms for combinatorial auctions with complement-free bidders. *Math. Oper. Res.* **35**(1), 1–13 (2010)
7. Dobzinski, S., Schapira, M.: An improved approximation algorithm for combinatorial auctions with submodular bidders. In: SODA, pp. 1064–1073 (2006)
8. Feldman, M., (Sef) Naor, J., Schwartz, R.: A unified continuous greedy algorithm for submodular maximization. In: FOCS, pp. 570–579 (2011)
9. Goel, G., Mehta, A.: Online budgeted matching in random input models with applications to adwords. In: SODA, pp. 982–991 (2008)
10. Khot, S., Lipton, R.J., Markakis, E., Mehta, A.: Inapproximability results for combinatorial auctions with submodular utility functions. *Algorithmica* **52**(1), 3–18 (2008)
11. Korula, N., Mirrokni, V.S., Zadimoghaddam, M.: Online submodular welfare maximization: greedy beats  $1/2$  in random order. *SIAM J. Comput.* **47**(3), 1056–1086 (2018)
12. Krause, A., Golovin, D.: Submodular function maximization, working paper (2014)
13. Lee, J., Mirrokni, V.S., Nagarajan, V., Sviridenko, M.: Maximizing nonmonotone submodular functions under matroid or knapsack constraints. *SIAM J. Discrete Math.* **23**(4), 2053–2078 (2010)
14. Mitrovic, M., Kazemi, E., Zadimoghaddam, M., Karbasi, A.: Data summarization at scale: a two-stage submodular approach. In: ICML, pp. 3593–3602 (2018)
15. Schrijver, A.: *Combinatorial Optimization-Polyhedra and Efficiency*. Springer, Berlin (2003)
16. Stan, S., Zadimoghaddam, M., Krause, A., Karbasi, A.: Probabilistic submodular maximization in sub-linear time. In: ICML, pp. 3241–3250 (2017)
17. Ward, J.: Oblivious and non-oblivious local search for combinatorial optimization, Ph. D. thesis, University of Toronto (2012)
18. Yang, R., Shuyang, G., Gao, C., Weili, W., Wang, H., Dachuan, X.: A constrained two-stage submodular maximization. *Theor. Comput. Sci.* **853**, 57–64 (2021)



# $(\mathbb{Z}, \text{succ}, U)$ , $(\mathbb{Z}, E, U)$ , and Their CSP's

William Gasarch , Michael Laskowski , and Shaopeng Zhu <sup>(✉)</sup> 

University of Maryland, College Park, USA  
{gasarch,laskow,szhu}@umd.edu

**Abstract.** When we expand the simple structure  $(\mathbb{Z}, \text{succ})$  with one unary predicate  $U$ , its CSP (Constraint Satisfaction Problem) may vary in complexity. We find some sufficient conditions for its tractability, prove bounds on its complexity, and then generalize our results to more complicated structures. We also give a Karp-equivalent characterization of  $\text{CSP}(\mathbb{Z}, \text{succ}, U)$ 's.

## 1 Introduction

In 1990, Hell and Nešetřil [17] showed that, fixing a finite undirected graph  $H$ , the decision problem “for finite input  $G$ , is there a map  $f : V(G) \rightarrow V(H)$ , s.t.  $\forall x, y((x, y) \in E(G) \implies (f(x), f(y)) \in E(H))$ ?” is in P if  $H$  is bipartite, NP-complete otherwise. Such adjacency-preserving maps are *graph homomorphisms*, and the problem is called *H-coloring*. The definition of *homomorphisms* naturally generalize to arbitrary relational structures by asserting each relation is unilaterally preserved by the map between domains. This gives rise to:

### Problem 1 (CSP( $H$ )).

**Parameter (Fixed) :**  $H := (D; R_1^D, \dots, R_k^D)$  a relational structure, with the symbol tuple  $(R_1, \dots, R_k)$  its signature.

**Input:** a finite  $(R_1, \dots, R_k)$ -structure  $G$ .

**Output:** does  $G$  homomorphically map to  $H$ ?

Such problems are called the *constraint satisfaction problems* (CSP's); we also use  $\text{CSP}(H)$  to denote the set of inputs that answers “Yes”. Correspondingly, the *Search-CSP*'s admit the same description, except on the “Yes” instances they output a homomorphism  $h : G \rightarrow H$ .

Seeing the Hell-Nešetřil's *H-coloring* dichotomy theorem, one naturally wonders if similar dichotomy exists for all finite relational structures. A dividing line was conjectured by Feder and Vardi [16]. Classification of many subclasses of finite domain CSP's (e.g. [1, 13, 15]) were studied over the years, but the conjecture remained open until independently proved by Bulatov and Zhuk ([14, 23, 24]) in 2017. A good reference for equivalent characterizations of the finite CSP dividing line can be found in [3].

For infinite domain CSP's, dichotomy has been established for many classes of “nice” structures; for example, fixing an infinite structure  $(A, \tau)$ , the class

$\{\mathfrak{A}' := (A, \tau') : R^{\mathfrak{A}'}$  is first order definable in  $\mathfrak{A}$  for each  $R \in \tau'\}$  is called *first order reducts* of  $\mathfrak{A}$ ; that is, all relational structures with the same domain  $A$  and whose finitely many relations are respectively definable in terms of  $\mathfrak{A}$ . Dichotomy has been established for, e.g., the first order reducts of the Rado graph [11], of unary structures [22], of  $(\mathbb{Q}, <)$  [5], and of countable (ultra)homogeneous graphs [10]. These structures are “nice” in that their first order theory has (up to isomorphism) a unique countable model, a property called  *$\omega$ -categoricity*. Another line of work focuses on dichotomizing the “numeric domains”, such as first order reducts of  $(\mathbb{Z}, \text{succ})$  [8], more generally of  $(\mathbb{Z}, <)$  [9], and of  $(\mathbb{Z}, +, 1)$  containing 1 [7]. A long standing open problem is to characterize first order reducts of  $(\mathbb{Z}, \leq, +)$ . For thorough introduction to these topics, see [3, 6].

So why do we care about characterizing  $\text{CSP}(\mathbb{Z}, \text{succ} := \{(x, y) : y = x + 1\}, U)$ , a specific type of CSP’s? On one hand, an efficient greedy algorithm solves  $\text{CSP}(\mathbb{Z}, \text{succ})$  (Sect. 2); on the other hand, with an infinite  $U \subseteq \mathbb{Z}$ ,  $\text{CSP}(\mathbb{Z}, \text{succ}, U, \mathbf{0})$  can be hard: take  $U$  to be the halting set, for example, then the reduction  $n \mapsto$  the direct  $(n + 1)$ -path  $\mathbf{0} \dots n$  witnesses the undecidability of  $\text{CSP}(\mathbb{Z}, \text{succ}, U, \mathbf{0})$ , and likewise for  $U$  a harder set say in  $\Pi_3 \setminus \Sigma_2$ . One naturally wonders how wild  $\text{CSP}(\mathbb{Z}, \text{succ}, U)$ ’s could be,  $(\mathbb{Z}, \text{succ}, U)$  being virtually “sandwiched” between  $(\mathbb{Z}, \text{succ})$  and  $(\mathbb{Z}, \text{succ}, U, \mathbf{0})$  for each  $U \subseteq \mathbb{Z}$ .

Moreover,  $(\mathbb{Z}, \text{succ})$  and  $(\mathbb{Z}, \text{succ}, U, \mathbf{0})$  are both non- $\omega$ -categorical: the former has 2-types  $\{(x, x + k)\}_{k < \omega}$ , while the latter is rigid. This indicates that non- $\omega$ -categoricity, a model-theoretic “wildness” bears no overall impact on the tractability of the structure’s CSP. Hence  $\{(\mathbb{Z}, \text{succ}, U)\}_{U \subseteq \mathbb{Z}}$  affords another entry point for classification of the CSP’s of non- $\omega$ -categorical structures.

As mentioned above, recent work ([8, 9]; see also [6]) established that for a structure  $\mathfrak{A}$  over  $\mathbb{Z}$  with finite relational signature, if each relation is first-order definable in  $(\mathbb{Z}, <)$  without parameters, then  $\text{CSP}(\mathfrak{A})$  is either in P or NP-complete. A dichotomy has also been established when the relations definable from unary structures [22].  $(\mathbb{Z}, \text{succ}, U)$  *however is in neither case*:  $\text{succ}$  is not first order definable from unary structures and the only 0-definable unaries in  $(\mathbb{Z}, <)$  are  $\emptyset$  and  $\mathbb{Z}$ . Therefore we prove a class of “nicely gapped” unary expansions of  $(\mathbb{Z}, \text{succ})$  has efficient CSP (Sect. 3, Theorem 1), and proceed to characterize all  $\text{CSP}(\mathbb{Z}, \text{succ}, U)$  by gaps in  $U$  (Sect. 4, Theorem 2). In particular, some  $\text{CSP}(\mathbb{Z}, \text{succ}, U)$  might be candidates for relatively natural NP-intermediate problems (Sect. 4, Corollary 1 and discussion).

The characterizations in Sections (3)(4), understandably, highly relies on the arithmetic nature of  $(\mathbb{Z}, \text{succ})$ , so it is natural to ask if they could be generalized to unary expansions of undirected graphs over  $\mathbb{Z}$ , i.e. the  $(\mathbb{Z}, E, U)$ ’s with  $E$  just an irreflexive, symmetric binary relation. We record some progress in this direction in Sect. 5, and give some lower bound of CSP complexity (up to direct limits) assuming reasonable structural properties in the underlying digraph. In Sect. 6 we discuss some possible future directions.



## 2 Preliminaries

Definition of *constraint satisfaction problems* and *homomorphisms* are introduced in Sect. 1. In e.g. Section 5, we adopt the alternative formulation:

**Problem 2 (CSP( $D, \mathcal{S}$ ), alternative formulation).**

**Parameters (Fixed):** A structure  $\mathfrak{A} = (A, \tau^{\mathfrak{A}})$  with  $\tau$  a finite relational signature.

**Input:** a first order sentence  $\psi := \exists x_1 \dots x_r R(x_1, \dots, x_r)$ , where  $R(x_1, \dots, x_r)$  is a finite conjunction of '='s and atomics in  $\tau$ .

**Output:** does  $\mathfrak{A} \models \psi$ ?

A walk through the definitions should convince one that these two formulations are equivalent up to the transformation  $\psi := \exists x_1, \dots, x_r R_1(\mathbf{x}) \wedge \dots R_j(\mathbf{x}) \mapsto$  the structure  $G$  on vertices  $\{x_1, \dots, x_r\}$  with each  $R_j^G(\bullet \in [1, j])$  as positively prescribed in  $\psi$ ,<sup>1</sup> and vice versa. First order sentences (resp. formulae) in the form of inputs to Problem 2 are *pp-sentence* (resp. *pp-formulae*). For  $\mathfrak{A} = (A, \tau^{\mathfrak{A}})$ , a relation  $R \subseteq A^r$  is *pp-definable* (resp. *fo-definable*) without parameters in  $\mathfrak{A}$  if there exists a pp(resp. first order)  $\tau$ -formula  $\psi(x_1, \dots, x_r)$  s.t.  $(a_1, \dots, a_r) \in R \iff \mathfrak{A} \models \psi(a_1, \dots, a_r)$ .

We use  $\leq_m^p$  (resp.  $\leq_T^p$ ) to denote the polynomial-time many-one, or *Karp* (resp. polynomial-time Turing, or *Cook*) reductions;  $\equiv_m^p$  (resp.  $\leq_T^p$ ) denotes the corresponding equivalence. The following is well known:

**Fact 1** (e.g. [3]). *Let relational structures  $\mathfrak{A} := (A, \tau^{\mathfrak{A}}), \mathfrak{A}' := (A, \zeta^{\mathfrak{A}'})$  be such that for each  $R \in \tau, R^{\mathfrak{A}}$  is pp-definable in  $\mathfrak{A}'$ . Then  $\text{CSP}(\mathfrak{A}) \leq_m^p \text{CSP}(\mathfrak{A}')$ .*

When describing structures  $(A, \tau^{\mathfrak{A}})$ , we drop the superscript if  $\mathfrak{A}$  is clear from context. E.g.,  $\mathfrak{A} = (\mathbb{Z}, \text{succ})$  is technically  $(\mathbb{Z}, \text{succ}^{\mathfrak{A}} := \{(x, y) : y = x + 1\})$ . Likewise for  $U \subseteq \mathbb{Z}, \mathfrak{A} := (\mathbb{Z}, \text{succ}, U)$  means the unary relation on in  $\mathfrak{A}$  is interpreted as  $U^{\mathfrak{A}} := U$ .

Both  $\text{CSP}(\mathbb{Z}, \text{succ})$  and  $\text{SEARCH-CSP}(\mathbb{Z}, \text{succ})$  can be solved by the same efficient greedy algorithm that, for each connected component  $\mathfrak{D}_c$  of the input  $\mathfrak{D}$ , starts by assigning an arbitrary vertex  $a \in D_c$  to 0 and then assigns each of  $a$ 's neighbor to  $-1$  (if  $(*, a) \in \text{succ}^{\mathfrak{D}}$ ) or  $1$  (if  $(a, *) \in \text{succ}^{\mathfrak{D}}$ ), then recurses on those neighbors. We refer to this algorithm as *the CSP( $\mathbb{Z}, \text{succ}$ ) decider / searcher*.

Let  $\mathfrak{A}$  be a  $\tau$ -structure;  $\text{Aut}(\mathfrak{A})$  (resp.  $\text{End}(\mathfrak{A})$ ) denotes its group of automorphisms (resp. monoid of endomorphisms).  $\mathfrak{A}$  is  $\omega$ -categorical if its first order theory  $\text{Th}(\mathfrak{A})$  (namely all sentences satisfied by  $\mathfrak{A}$ ) is  $\omega$ -categorical, i.e.  $\mathfrak{A}' \models \text{Th}(\mathfrak{A}) \wedge |A'| = \aleph_0 \implies \mathfrak{A}' \cong \mathfrak{A}$ . Another well known fact used in Sect. 5:

**Fact 2** (e.g. [19], equivalent characterization of  $\omega$ -categoricity). *Let  $\tau$  be a countable signature,  $T$  a complete  $\tau$ -theory which has infinite models. Then the following are equivalent:*

<sup>1</sup> That is, for the  $j$ -ary relation  $R_j$ , each  $(y_1, \dots, y_j) \in \{x_1, \dots, x_r\}^j$  is in  $R_j^G \iff R_j(y_1, \dots, y_j)$  is a conjunct appearing in  $\psi$ .

1. All countable models of  $T$  are isomorphic;
2. Let  $\mathfrak{A} \models T$ , then  $\text{Aut}(\mathfrak{A})$  has finitely many orbits in its action  $(a_1, \dots, a_n) \mapsto (ga_1, \dots, ga_n)$  on  $A^n$ , for any  $n \geq 1$ .

### 3 A Sufficient Condition for Efficiency of $\text{CSP}(\mathbb{Z}, \text{succ}, U)$

We do not believe there is a many-one or Turing reduction from  $U$  to  $\text{CSP}(\mathbb{Z}, \text{succ}, U)$ ; intuitively, without a  $\mathbf{0}$  (or any constant for that matter) to “anchor the origin”,  $\text{End}(\mathbb{Z}, \text{succ}) = \text{Aut}(\mathbb{Z}, \text{succ}) \cong \mathbb{Z}$  and only the gap patterns in  $U$  decide whether a particular translation of  $\{\text{succ}\}$ -homomorphism makes a  $\{\text{succ}, U\}$ -homomorphism. In Sect. 4 we shall formalize this intuition; here, we give some nice conditions on the gaps to guarantee the tractability of  $\text{CSP}(\mathbb{Z}, \text{succ}, U)$ . First, things are nice if  $U$  is periodical:

**Observation 1.** *Let  $U = a\mathbb{Z} + b$  for some  $a, b \in \mathbb{Z}$ . Then  $\text{CSP}(\mathbb{Z}, \text{succ}, U) \in \text{P}$ .*

*Proof.* By a slight modification to the  $\text{SEARCH-CSP}(\mathbb{Z}, \text{succ})$  solver. Namely, for each component of input  $\mathfrak{D}$ , if it has some  $v \in U^{\mathfrak{D}}$ , initiate the  $\text{SEARCH-CSP}(\mathbb{Z}, \text{succ})$  solver by assigning  $h(v) := b$ . Correctness follows from the fact that  $\text{Aut}(\mathbb{Z}, \text{succ}, U) \cong \{ak : k \in \mathbb{Z}\}$ . For full details, see Appendix C.1.  $\square$

$\text{CSP}(\mathbb{Z}, \text{succ}, U)$  also behaves tamely, with its hardness upper-bounded by  $U$ , if the gaps in  $U$  are eventually large. Let us generalize this observation.

**Definition 1.** *Let  $U = \{u(i)\}_{i < \omega} \subseteq \mathbb{Z}$ .  $U$  is eventually largely gapped (ELG) if  $\exists c > 0$ , s.t.  $\forall n > c, \forall n' \in \omega \setminus \{n\}, |u(n) - u(n')| > n$ .*

**Example 1.** *Let  $f(x) \in \mathbb{Z}[x]$  with  $\deg(f(x)) \geq 2$ , e.g.  $6x^3 + 4x + 7$ . One may verify that  $\underline{U}_f := \{f(x) : x < \omega\}$  is ELG.<sup>2</sup>*

**Example 2.** *Likewise for any differentiable  $g(x) : \mathbb{R} \rightarrow \mathbb{R}$ , if  $g(\omega) \subseteq \mathbb{Z}$  with superpolynomial growth rate and some more mild assumptions, we have  $\underline{U}_g := \{g(x) : x < \omega\}$  is ELG. E.g.  $g(x) = -x \cdot 2^x$ .<sup>3</sup>*

**Definition 2.** *Let  $U = \{u(i)\}_{i < \omega} \subseteq \mathbb{Z}$ .  $U$  has small representation (SR) if there exists a  $p(x) \in \mathbb{N}[x]$  s.t.  $|\langle u(i) \rangle| \sim O(p(i))$  for each  $i < \omega$ .<sup>4</sup> we say  $U$  is ELG-SR if  $U$  is ELG and has SR.*

**Example 3.** *In Examples 1 and 2, both  $U_f$  and  $U_g$  have SR, and are hence ELG-SR. Note  $|\langle -2^n \rangle| \sim O(n)$ .*

<sup>2</sup> As  $x \rightarrow \infty$ ,  $f$  is eventually monotonic, and the formal derivative  $|f'(x)| \sim \Omega(x)$ , and when  $|f'(x)| \sim \Theta(x)$  the absolute value of coefficient of  $x$ -term is at least 2. For  $c \gg 0$  we have  $|f(x) - f(x^*)| \geq \max\{|f(x) - f(x-1)|, |f(x) - f(x+1)|\} > x$  whenever  $c < x \in \omega, x^* \in \omega \setminus \{x\}$ . So indeed  $U_f$  is ELG.

<sup>3</sup> The two more mild assumptions are (1)  $g$  is eventually monotonic and (2)  $|g'(x)| \sim \omega(|x|)$ . Indeed, for  $x \gg 0$  we have  $\frac{|g(x) - g(x-1)|}{1} \geq |g'(x-1)| \sim \omega(x-1)$  so  $LHS > x$  eventually, and similarly,  $|g(x+1) - g(x)| > x$  eventually.

<sup>4</sup> Conventionally,  $\langle \bullet \rangle$  is the binary representation of an integer and  $|\bullet|$  is the length.

**Proposition 1.** *Let  $U = \{u(i)\}_{i < \omega}$  be ELG-SR. Then  $\text{CSP}(\mathbb{Z}, \text{succ}, U) \leq_T^p U$ .*

*Proof.* For each connected component  $\mathfrak{D}_c$  of the input  $\mathfrak{D} = (D, \text{succ}^{\mathfrak{D}}, U^{\mathfrak{D}})$ , reject if the  $\text{CSP}(\mathbb{Z}, \text{succ})$ -solver rejects on  $\mathfrak{D}_c$ ; otherwise the solver finds a succ-homomorphism  $h : (\mathfrak{D}_c, \text{succ}^{\mathfrak{D}} \cap D_c^2) \rightarrow (\mathbb{Z}, \text{succ})$ . Thanks to ELG, one needs to shift  $h$  at most  $|D_c|$  times to decide if any translation makes  $h$  a  $(\text{succ}, U)$ -homomorphism. SR ensures the absence of exponential blowup in this algorithm. For full details, see Appendix A.1 .  $\square$

**Corollary 1.** *If  $U$  is ELG-SR,  $\text{CSP}(\mathbb{Z}, \text{succ}, U)$  is in P.*

*Proof.* Proposition 1 puts  $\text{CSP}(\mathbb{Z}, \text{succ}, U) \in \text{P}^U$ ; furthermore each  $U$ -oracle call above further reduces to linearly many comparisons of two polynomial-sized numbers, generating polynomial overhead. For details, see Appendix A.1 .  $\square$

Proposition 1 generalizes to expansion of  $(\mathbb{Z}, \text{succ})$  by finitely many unaries. As a caveat, one needs to handle the “cross-set” gaps and work with the “least upper bound” of the finite collection of unaries in terms of hardness.

**Definition 3.** *Let  $k \geq 1$ . A collection  $\mathcal{U} := \{U_1 = \{u(i, 1)\}_{i < \omega}, \dots, U_k = \{u(i, k)\}_{i < \omega} \subseteq \mathbb{Z}\}$  is eventually mutually largely gapped (EMLG) if*

1. Each  $U_j$  ( $j \in [k]$ ) is ELG, and
2. For each  $(j, l) \in \binom{[k]}{2}$ ,  $\exists c_{j,l} > 0$ , s.t.
  - $\forall n_j > c_{j,l}, \forall n_l < \omega, |u(n_j, j) - u(n_l, l)| > n_j$ , and
  - $\forall n_l > c_{j,l}, \forall n_j < \omega, |u(n_l, l) - u(n_j, j)| > n_l$ .

Say  $\mathcal{U}$  is EMLG-SR if it is EMLG and each  $U_j$  ( $j \in [k]$ ) has SR.

**Example 4.** *For differentiable functions  $f_1, f_2 \in \mathbb{R}^{\mathbb{R}}$  each as described in Example 2, if they have opposite signs when restricted to  $\omega$ ,  $\{U_{f_1}, U_{f_2}\}$  is EMLG; in particular e.g. for  $f_1(x) = x^3, f_2(x) = -2^x$ ,  $\{U_{f_1}, U_{f_2}\}$  is EMLG-SR.*

**Notation 1.** *Let  $U_1, \dots, U_k \subseteq \mathbb{Z}$ . We denote by  $\varinjlim_{j \in [k]} U_j$  the following set:*

$$\varinjlim_{j \in [k]} U_j := \{n = mk + (r \bmod k) : r \in [1, k], m \in U_r\} \quad (1)$$

Note that  $\varinjlim_{j \in [k]} U_j$  is just encoding  $U_j$ 's into one set s.t. each  $U_j$  reduces to it, and it is the “easiest” such set in the sense of Karp reductions.<sup>5</sup> Now we generalize Proposition 1 and Corollary 1 below. Both proofs resemble and generalize the single-unary case, whose details are put in Appendix A.2.

**Theorem 1.** *Let  $\{U_j\}_{j \in [k]}$  be EMLG-SR. Then  $\text{CSP}(\mathbb{Z}, \text{succ}, (U_j)_j) \leq_T^p \varinjlim_j U_j$ .*

**Corollary 2.** *If  $\{U_j\}_j$  is EMLG-SR, Then  $\text{CSP}(\mathbb{Z}, \text{succ}, U_1, \dots, U_k) \in \text{P}$ .*

<sup>5</sup> Appendix C.2.

## 4 Bounds and Characterization of $\text{CSP}(\mathbb{Z}, \text{succ}, U)$

### 4.1 “Lower Bounding” $\text{CSP}(\mathbb{Z}, \text{succ}, U)$

Let  $U \subseteq \mathbb{Z}$  be arbitrary. Recall  $\text{CSP}(\mathbb{Z}, \text{succ}, U) \leq_m^p \text{CSP}(\mathbb{Z}, \text{succ}, U, \mathbf{0})$  as the upper bound. It would be nice if the converse held, but we believe it is unlikely due to similar “anchoring” issues (See e.g. Section 3 and *infra*). However,

**Proposition 2.**  $\text{CSP}(\mathbb{Z}, \text{succ}, U, \mathbf{0}) \leq_T^p \varinjlim \{U, \text{CSP}(\mathbb{Z}, \text{succ}, U)\}$ .

*Proof.* Test the non- $\mathbf{0}$  components of the input by  $\text{CSP}(\mathbb{Z}, \text{succ}, U)$ -oracle calls. For the  $\mathbf{0}$ -component, find if a  $\{\text{succ}, \mathbf{0}\}$ -homomorphism  $h$  exists, and make  $O(|D|)$ -calls to the  $U$ -oracle to find if  $h$  is a  $\{\text{succ}, U, \mathbf{0}\}$ -homomorphism. For details, see Appendix A.3.  $\square$

Hence, up to taking direct limit with  $U$ ,  $\text{CSP}(\mathbb{Z}, \text{succ}, U)$  is “lower bounded” by  $\text{CSP}(\mathbb{Z}, \text{succ}, U, \mathbf{0})$  in the  $\leq_T^p$ -order. The following is a notable corollary with proof in Appendix A.3.

**Corollary 3.** *If  $U \in \mathcal{P}$ , or  $U \leq_T^p \text{CSP}(\mathbb{Z}, \text{succ}, U)$ , then  $\text{CSP}(\mathbb{Z}, \text{succ}, U) \equiv_T^p \text{CSP}(\mathbb{Z}, \text{succ}, U, \mathbf{0})$ .*

### 4.2 Karp-Equivalent Characterizations of $\text{CSP}(\mathbb{Z}, \text{succ}, U)$

Recall (e.g. Sect. 3) that intuitively, for any  $U \subseteq \mathbb{Z}$  its gap patterns determine the members of  $\text{CSP}(\mathbb{Z}, \text{succ}, U)$ , and conversely one may recover the gap patterns of  $U$  from  $\text{CSP}(\mathbb{Z}, \text{succ}, U)$ . Our characterization below formalizes this intuition.

**Definition 4.** *Let  $U \subseteq \mathbb{Z}$  with  $|U| \geq 2$ . Define  $\text{Gap}(U) :=$*

$$\{(n, S_1, S_2) \in (\omega \setminus \{0, 1\}) \times 2^{\binom{n}{2}} \times [-n, n]^{S_1} \mid \exists t \in U^n, (\forall (i < j) \in S_1)(t(j) - t(i) = S_2(i, j))\}$$

Informally  $\text{Gap}(U)$  is the collection of gap patterns realized by some  $U$ -tuple. Note, though, that having  $S_1 \subseteq \binom{n}{2}$  is crucial: we need to include the cases where only *some* distances between the  $n$  vertices are specified. This corresponds to the fact that homomorphisms only remember adjacencies but not non-adjacencies, and affords the following equivalence:

**Theorem 2.** *Let  $U \subseteq \mathbb{Z}$  with  $|U| \geq 2$ . Then  $\text{Gap}(U) \equiv_m^p \text{CSP}(\mathbb{Z}, \text{succ}, U)$ .*

*Proof.* On a  $\text{Gap}(U)$ -instance  $(n, S_1, S_2)$ , construct  $\mathfrak{D}$  as follows: start with  $D = U^{\mathfrak{D}} = \{1, \dots, n\}$ , then add paths of length  $S_2(i, j)$  between vertices  $i, j$  for each  $(i < j) \in S_1$ . Conversely, on a  $\text{CSP}(\mathbb{Z}, \text{succ}, U)$ -instance  $\mathfrak{D} = (D, \text{succ}^{\mathfrak{D}}, U^{\mathfrak{D}})$ , first let the subgraph  $\mathfrak{D}'$  collect each  $\mathfrak{D}$ -component that intersects with  $U^{\mathfrak{D}}$  at more than 1 vertices; let  $n := |\mathfrak{D}' \cap U^{\mathfrak{D}}|$ , and let  $(S_1, S_2)$  keep track of the positive finite distances of each  $U^{\mathfrak{D}}$ -pair that lies on the same component of  $\mathfrak{D}'$ . For full details, See Appendix A.4.  $\square$

We close the section with an interesting question:

**Question 1.** *Let  $V \subseteq \omega$  be infinite. Is there a  $U_V \subseteq \mathbb{Z}$  s.t.  $V \leq_T \text{Gap}(U_V)$ ?*

That is, we would like for any infinite set  $V$  of naturals, there exists a set  $U_V$  whose “gap patterns” decide  $V$ -membership. Assuming that, an immediate corollary would be the unboundedness of computational complexity of  $\text{Gap}(U)$  and  $\text{CSP}(\mathbb{Z}, \text{succ}, U)$ 's in general. If, in addition,  $\leq_T$  in Question 1 could be refined to  $\equiv_m^p$ , then Ladner's theorem [20] and a recent construction [3, 4] would imply the existence of NP-intermediate CSP's and CoNP-intermediate CSP's respectively of the form  $\text{CSP}(\mathbb{Z}, \text{succ}, U)$ .

We summarize our results in Fig. 2, Appendix B.1.

## 5 From $(\mathbb{Z}, \text{succ}, U)$ to $(\mathbb{Z}, E, U)$

Now we generalize our results in Sections (3)(4) into arbitrary *partially-1-colored digraphs* on  $\mathbb{Z}$ , i.e. structures of the form  $(\mathbb{Z}, E, U)$  with  $E \subseteq \mathbb{Z}^2, U \subseteq \mathbb{Z}$ . With due care, results here can be further generalized: see remark in Appendix C.7.

As a motivating example, fix any  $d \geq 1$  and consider the relation  $\text{Diff}_d(x, y) \iff y = x + d$ . Note that  $\text{Diff}_d$  is pp-definable in  $(\mathbb{Z}, \text{succ})$ . The digraph  $(\mathbb{Z}, \text{Diff}_d)$  consists of  $d$  pairwise isomorphic components, each isomorphic to  $(\mathbb{Z}, \text{succ})$ , therefore  $\text{CSP}(\mathbb{Z}, \text{succ}) = \text{CSP}(\mathbb{Z}, \text{Diff}_d) \in \text{P}$ . Moreover, just as fixing one vertex fixes a homomorphism – if any exists – to  $(\mathbb{Z}, \text{succ})$ , for finite *connected*  $\{\text{Diff}_d, \mathbf{0}\}$ -structure  $\mathfrak{D}$  we have

$$\mathfrak{D} \in \text{CSP}(\mathbb{Z}, \text{Diff}_d, \mathbf{0}) \iff |\text{Hom}_{\text{Diff}_d, \mathbf{0}}(\mathfrak{D}, (\mathbb{Z}, \text{Diff}_d, \mathbf{0}))| = 1 \tag{2}$$

Further expanding the structure by  $U \subseteq \mathbb{Z}$ , we no longer have  $\text{CSP}(\mathbb{Z}, \text{succ}, U) = \text{CSP}(\mathbb{Z}, \text{Diff}_d, U)$ : the gaps are not preserved by the homomorphic equivalence between base structures. However by an argument similar to Sect. 3 Proposition 2, one may obtain:

**Observation 2.** *Let  $U \subset \mathbb{Z}$ . Then  $\text{CSP}(\mathbb{Z}, \text{Diff}_d, U, \mathbf{0}) \leq_T^p \varinjlim \{U, \text{CSP}(\mathbb{Z}, \text{Diff}_d, U)\}$ .*

Details and more discussions on generalizing results in Sect. 3 to  $(\mathbb{Z}, \text{Diff}_d, U)$  can be found in Appendix C.3.

Now consider any partially-1-colored digraph  $\mathfrak{A} := (\mathbb{Z}, E, U)$ . We are interested in generalizing Sect. 3 results to complexity lower bounds of  $\text{CSP}(\mathfrak{A})$ .

**Definition 5.** *Let  $U \subseteq \mathbb{Z}, E \subseteq \mathbb{Z}^2$ , and  $\mathfrak{A} := (\mathbb{Z}, E^{\mathfrak{A}} := E, U^{\mathfrak{A}} := U)$ . Consider its reduct  $\mathfrak{A}^\perp := (\mathbb{Z}, E)$ . Let  $\mathbf{0}$  be a constant.<sup>6</sup> We say  $\mathfrak{A}^\perp$  is*

1. pointed homomorphically rigid (PHR) if if for any finite, connected  $\{E, \mathbf{0}\}$ -structure  $\mathfrak{D}$ ,

$$\mathfrak{D} \in \text{CSP}(\mathfrak{A}^\perp, \mathbf{0}) \iff |\text{Hom}_{\{E, \mathbf{0}\}}(\mathfrak{D}, (\mathfrak{A}^\perp, \mathbf{0}))| = 1 \tag{3}$$

<sup>6</sup> Any interpretation of  $\mathbf{0}$  works; without loss of generality we pick  $\mathbf{0}^{(\mathfrak{A}, \mathbf{0})} = 0 \in \mathbb{Z}$ .

2. freely translational (FT) if  $\text{Aut}_E(\mathfrak{A}^\perp)$  contains all translations  $f_z : x \mapsto x + z$  ( $z \in \mathbb{Z}$ ).

**Theorem 3.** Let  $\mathfrak{A} := (\mathbb{Z}, E, U)$  be such that  $\mathfrak{A}^\perp$  is PHR and FT. Then,<sup>7</sup>

$$\text{CSP}(\mathfrak{A}, \mathbf{0}) \leq_T^p \varinjlim \{U, \text{CSP}(\mathfrak{A}), \text{SEARCH-CSP}(\mathfrak{A}^\perp)\} \quad (4)$$

*Proof.* Similar to Proposition 2. Decompose input  $\mathfrak{D} = (\mathbb{Z}, E^\mathfrak{D}, U^\mathfrak{D}, \mathbf{0}^\mathfrak{D})$  into connected components. For components  $\mathfrak{D}_b$  avoiding  $\mathbf{0}^\mathfrak{D}$ , call the  $\text{CSP}(\mathfrak{A})$  oracle.

For  $\mathfrak{D}_0 \ni \mathbf{0}^\mathfrak{D}$ , find an  $\{E\}$ -homomorphism  $h : \mathfrak{D}_0 \rightarrow (\mathfrak{A}^\perp)$  if it exists, by calling  $\text{SEARCH-CSP}(\mathfrak{A}^\perp)$ . By FT,  $s : x \mapsto x - h(\mathbf{0}^\mathfrak{D})$  is an  $E$ -endomorphism, hence  $s \circ h$  is an “pointed”  $E$ -homomorphism where  $\mathbf{0}^\mathfrak{D} \mapsto \mathbf{0} = \mathbf{0}^\mathfrak{A}$ , i.e. an  $\{E, \mathbf{0}\}$ -homomorphism. By PHR,  $s \circ h$  is the unique  $\{E, \mathbf{0}\}$ -homomorphism  $\mathfrak{D}_0 \rightarrow (\mathfrak{A}^\perp, \mathbf{0})$ . Now to check if  $s \circ h$  is further an  $\{E, \mathbf{0}, U\}$ -homomorphism, simply test if  $s \circ h(u) \in U = U^\mathfrak{A}$  for each  $u \in U^\mathfrak{D} \cap D_0$ .<sup>8</sup>  $\square$

Theorem 3 says that when the underlying digraph  $\mathfrak{A}^\perp = (\mathbb{Z}, E)$  has nice algebraic properties (PHR-FT), one obtains a generalized lowerbound of  $\text{CSP}(\mathfrak{A})$  “up to direct limit” given by  $\text{CSP}$  of the pointed structure  $(\mathfrak{A}, \mathbf{0})$ . This also has some structural implications:

**Proposition 3.** For  $(\mathfrak{A}^\perp, \mathbf{0}) = (\mathbb{Z}, E, \mathbf{0})$ , let  $\mathfrak{A}^\perp_0$  denote its central component, i.e. the pointed component of the underlying digraph containing  $\mathbf{0}$ .

1. If  $\mathfrak{A}^\perp$  is PHR, then  $\text{End}_{E, \mathbf{0}}(\mathfrak{A}^\perp_0) = \{\mathbf{1}\}$ . The converse is false.
2. If  $\mathfrak{A}^\perp$  is FT, then  $\mathfrak{A}^\perp_0$  has infinite domain.
3. If  $\mathfrak{A}^\perp$  is PHR-FT, then  $(\mathfrak{A}^\perp, \mathbf{0})$  is not  $\omega$ -categorical, unless  $E = \emptyset$ .<sup>9</sup>

*Proof.* 1. Assume for contradiction that  $\text{End}_{E, \mathbf{0}}(\mathfrak{A}^\perp_0) \neq \{\mathbf{1}\}$ , so there exists  $\sigma \in \text{End}_{E, \mathbf{0}}(\mathfrak{A}^\perp_0)$  and  $a \neq b \in \mathfrak{A}^\perp_0$  s.t.  $\sigma(a) = b$ . Note that  $a \neq \mathbf{0}$  for  $\sigma$  preserves  $\mathbf{0}$ . By connectedness, there exists a finite signed path  $\mathbf{0} \dots a \subseteq \mathfrak{A}^\perp_0$ . Take  $\mathfrak{D} := \mathfrak{A}^\perp_0[\{\mathbf{0}, \dots, a\}]$ , which is again connected thanks to that signed path. Let  $h_1 : \mathfrak{D} \hookrightarrow \mathfrak{A}^\perp_0$  the embedding. Now,  $h_2 := \sigma \circ h_1 \in \text{Hom}_{E, \mathbf{0}}(\mathfrak{D}, \mathfrak{A}^\perp_0)$  with  $b = h_2(a) \neq h_1(a) = a$ , contradicting PHR. Falsity of the converse is given by a counter-example in Appendix C.4.

<sup>7</sup> Note: when  $E = \text{succ}$ , we have  $\text{SEARCH-CSP}(\mathbb{Z}, E) \leq_T^p \text{CSP}(\mathbb{Z}, E) \in \mathbf{P}$ , so Theorem 3 gives back Proposition 2. In general we do not know if the Search to Decision reduction works for *infinite* structures.

<sup>8</sup> Note that the runtime of this SEARCH-CSP accounts for the time to write down the  $h(x)$ 's for each  $x \in \mathfrak{D}$ . Later when we compute the shift  $a - h(\mathbf{0}^\mathfrak{D})$ , the overhead is dominated by the runtime of the SEARCH-CSP. The total overhead of computing  $s \circ h$  is therefore at most that of running  $O(|D|)$  times of the SEARCH-CSP cost.

<sup>9</sup> Some remark: first, Proposition 3(3) implies  $(\mathfrak{A}, \mathbf{0})$  is also not  $\omega$ -categorical. Also, if  $E = \emptyset$  then  $(\mathfrak{A}, \mathbf{0})$  is indeed  $\omega$ -categorical. We use the following equivalent characterization: a structure  $\mathfrak{A}$  is  $\omega$ -categorical iff Some countable model of  $T$  realizes only finitely many complete  $n$ -types for each  $n < \omega$ . For any  $n \geq 1$ , a complete  $n$ -type  $(x_1, \dots, x_n)$  only needs to make a decision for each  $x_i$  as to whether  $x_i \in U$ , and whether  $x_i = \mathbf{0}$ . Thus there are finitely many complete  $n$ -types for any  $n$ . Likewise  $(\mathfrak{A}^\perp, \mathbf{0})$  is  $\omega$ -categorical for having even fewer  $n$ -types.

2. Assume  $E \neq \emptyset$  and assume for the sake of contradiction that  $|A^\downarrow_0| < \aleph_0$ .
  - (a) If  $|A^\downarrow_0| = 1$ , take  $(a_1, a_2) \in E$ . Note that  $a_1, a_2 \neq \mathbf{0}$  since  $\mathfrak{A}^\downarrow$  is isolated. The translation  $\sigma : x \mapsto x - a_1$  fails to be an  $\{E\}$ -endomorphism, as  $(\sigma(a_1), \sigma(a_2)) \notin E^{\mathfrak{A}^\downarrow}$ . This contradicts FT.
  - (b) Otherwise  $\aleph_0 > |A^\downarrow_0| \geq 2$ . Let  $b \in A^\downarrow_0 \setminus \{\mathbf{0}\}$  be of maximal absolute value in  $A^\downarrow_0$ . As  $\mathfrak{A}^\downarrow$  has FT and endomorphisms preserve connectedness, by repeatedly applying the translation  $x \mapsto x + b$  we obtain that  $\{kb\}_{k < \omega} \subset \mathfrak{A}^\downarrow_0$ , contradicting  $\aleph_0 > |A^\downarrow_0|$ .
3. We have  $\text{End}_{E, \mathbf{0}}(\mathfrak{A}^\downarrow_0) = \{\mathbf{1}\}$  and  $|A^\downarrow_0| = \aleph_0$ . Now  $\text{Aut}_{E, \mathbf{0}}(\mathfrak{A}^\downarrow, \mathbf{0}) \subseteq \text{Aut}_{E, \mathbf{0}}(\mathfrak{A}^\downarrow_0) \subseteq \text{End}_{E, \mathbf{0}}(\mathfrak{A}^\downarrow_0) = \{\mathbf{1}\}$ , so  $(\mathfrak{A}^\downarrow, \mathbf{0}) = (\mathbb{Z}, E, \mathbf{0})$  is rigid, and  $\text{Aut}_{E, \mathbf{0}}(\mathfrak{A}^\downarrow, \mathbf{0})$  on  $\mathbb{Z}$  has infinitely many orbits.  $(\mathfrak{A}, \mathbf{0})$  being an expansion of  $(\mathfrak{A}^\downarrow, \mathbf{0})$  is also rigid.

Proposition 3 (2) implies that, when  $|A^\downarrow_0| < \aleph_0$ , we need a different condition to guarantee a generalized lowerbound similar to Sect. 3.

**Definition 6.** Let  $\mathfrak{A} = (\mathbb{Z}, E, U)$  so  $(\mathfrak{A}^\downarrow, \mathbf{0}) = (\mathbb{Z}, E, \mathbf{0})$  with  $\mathfrak{A}^\downarrow_0$  its the central component. Say  $\mathfrak{A}^\downarrow$  has Transitive Endomorphism-action on Hom sets (TEH) if for any finite, connected  $E$ -structure  $\mathfrak{D}$ ,  $\text{End}_E(\mathfrak{A}^\downarrow[A^\downarrow_0])$  acts transitively on  $\text{Hom}_E(\mathfrak{D}, \mathfrak{A}^\downarrow[A^\downarrow_0])$  by post-composition.

Note that  $\mathfrak{A}^\downarrow[A^\downarrow_0]$  is the substructure of  $\mathfrak{A}^\downarrow$  induced on  $\mathfrak{A}^\downarrow_0$ 's central component.

**Proposition 4.** If  $|A^\downarrow_0| < \aleph_0$  and  $\mathfrak{A}^\downarrow$  has TEH, then:

$$\text{CSP}(\mathfrak{A}, \mathbf{0}) \leq_T^p \varinjlim \{U, \text{CSP}(\mathfrak{A}), \text{CSP}(\mathfrak{A}^\downarrow[A^\downarrow_0])\} \tag{5}$$

Changing  $\text{CSP}(\mathfrak{A}^\downarrow[A^\downarrow_0])$  to  $\text{SEARCH-CSP}(\mathfrak{A}^\downarrow[A^\downarrow_0])$  would be correct and tighter, but we use the following fact to get the current more succinct form :

**Fact 3 (e.g. Exercise in [2]).** Let  $\mathfrak{A}$  be a finite relational structure. Then  $\text{SEARCH-CSP}(\mathfrak{A}) \leq_T^p \text{CSP}(\mathfrak{A})$ .

Proof of Fact 3 is an exercise; proof of Proposition 4 is similar to Theorem 3, letting TEH play the role of FT. Details of both are included in Appendix A.5.

**Corollary 4.** Let  $U \in P$ .

1. If  $\mathfrak{A}^\downarrow = (\mathbb{Z}, E)$  is PHR-FT and if  $\text{SEARCH-CSP}(\mathfrak{A}^\downarrow) \leq_T^p \text{CSP}(\mathfrak{A}^\downarrow)$ , OR
2. If  $|A^\downarrow_0| < \aleph_0$ ,  $\mathfrak{A}^\downarrow$  has TEH, and if  $\text{CSP}(\mathfrak{A}^\downarrow[A^\downarrow_0]) \leq_T^p \text{CSP}(\mathfrak{A}^\downarrow)$ ,

then  $\text{CSP}(\mathfrak{A}, \mathbf{0}) \equiv_T^p \text{CSP}(\mathfrak{A})$ .

For verification of Corollary 4, see Appendix C.6. We summarize our main results in Fig. 3, Appendix B.2.

## 6 Future Directions

In Sects. 3 and 4, we studied how properties of  $U$  impacts the complexity of  $\text{CSP}(\mathbb{Z}, \text{succ}, U)$ . In particular, Sect. 3 identified some properties for  $U$  to guarantee a relatively tame behavior of  $\text{CSP}(\mathbb{Z}, \text{succ}, U)$ , while Sect. 4 characterized  $\text{CSP}(\mathbb{Z}, \text{succ}, U)$  by gaps in  $U$ . Some questions that initially motivated our research in that direction remains open:

**Problem 3.** *What are the equivalent conditions for  $\text{CSP}(\mathbb{Z}, \text{succ}, U)$  to be tractable? Are there  $U$ 's such that  $\text{CSP}(\mathbb{Z}, \text{succ}, U)$  is NP-intermediate or coNP-intermediate, assuming  $P \neq \text{NP}$ ? In particular, is the answer to Question 1 yes and can  $\leq_T$  there be refined to  $\equiv_m^p$ ?*

In Sect. 5 we generalized our lowerbound results to arbitrary  $(\mathbb{Z}, E, U)$  where the underlying digraph has nice properties. Some next steps may include:

1. Study if with reasonable properties on  $E$ , the  $\varinjlim()$  in the quasi-lowerbound could be removed.
2. Refine the characterization of complexity of  $\text{CSP}(\mathbb{Z}, E, U)$  by properties on  $E$  and  $U$ . Some discussion on one specific direction of such refinement is attached in Appendix C.8.

**Acknowledgement.** Partially supported by NSF grant DMS-1855789. We thank the reviewers for their time and enlightening comments. The third author thanks the first two authors for their great advice.

## A Proof Details of Interesting Claims

### A.1 Details of Section 3, Proposition 1 and Corollary 1

First, we show the details of the proof that if  $U = \{u(i)\}_{i < \omega}$  is ELG-SR, then  $\text{CSP}(\mathbb{Z}, \text{succ}, U) \leq_T^p U$ . On  $\mathfrak{D} = (D, \text{succ}^{\mathfrak{D}}, U^{\mathfrak{D}})$ , for each component  $\mathfrak{D}_e$ :

1. If  $D_e \cap U^{\mathfrak{D}} = \emptyset$ , run  $\text{CSP}(\mathbb{Z}, \text{succ})$  decider on  $(D_e, \text{succ}^{\mathfrak{D}} \cap D_e^2)$ ; if  $|D_e \cap U^{\mathfrak{D}}| = 1$ , run the  $\text{CSP}(\mathbb{Z}, \text{succ})$  algorithm beginning with assigning the unique  $d_e \in D_e \cap U^{\mathfrak{D}}$  to an arbitrary  $b \in U$ .

Correctness follows from the observation that, in these cases we are still free to translate a  $\text{succ}$ -homomorphism as long as one exists.

2. Now  $|D_e \cap U^{\mathfrak{D}}| \geq 2$ . Since  $U$  is ELG, by definition there exists  $c \in \omega$  witnessing it, i.e. gaps in  $U$  are large after index  $c$ . Pick arbitrary  $d_e \in D_e \cap U^{\mathfrak{D}}$ . Run the following:



```

while  $i \leq \max\{c, |D_e|\}$  do
  run CSP( $\mathbb{Z}, \text{succ}$ ) solver on  $\mathfrak{D}_e$  initiated by sending  $d_e$  to  $u(i)$ 
  if CSP( $\mathbb{Z}, \text{succ}$ ) rejects then
    reject
  else
    //CSP( $\mathbb{Z}, \text{succ}$ ) found  $h \in \text{Hom}(\mathfrak{D}_e, (\mathbb{Z}, \text{succ}))$  with
     $h(d_e) = u(i)$ 
    for  $d'_e \in D_e \cap U^{\mathfrak{D}}$  do
      | Test if  $h(d'_e) \in U$ , reject if not
    end
    accept for  $\mathfrak{D}_e$ 
  end
end

```

**Algorithm 1:** The reduction  $\text{CSP}(\mathbb{Z}, \text{succ}, U) \leq_T^p U$ .

And  $\mathfrak{D}$  is accepted if and only if each  $\mathfrak{D}_e$  is accepted. To see efficiency: since  $U$  has SR,  $|\langle u(i) \rangle|$  is polynomial in  $i$ , and  $i$  is linear in  $|D_e| \leq |D|$ . If a  $\{\text{succ}\}$ -homomorphism  $h$  is found, we have polynomially many calls to the  $U$ -oracle; note that each  $h(d'_e) \in [u(i) - n, u(i) + n]$ , so it remains polynomial size. The whole algorithm is therefore poly-calls to  $U$  with poly-overhead.

To see correctness, first observe the following fact: in a connected component of size  $n$ , for any vertices  $x \neq y$  the longest “signed path” (vertices  $x_0 := x, x_1, \dots, x_k := y$  s.t. for each  $i$ , either  $(x_i, x_{i+1}) \in E$  or  $(x_{i+1}, x_i) \in E$ ) has at most  $n - 1$  edges, and the signed length (computed by adding 1 if  $(x_i, x_{i+1}) \in E$  and subtracting 1 if  $(x_{i+1}, x_i) \in E$ ).<sup>10</sup> is at most  $n - 1$ . Whenever the component homomorphically maps to  $(\mathbb{Z}, \text{succ})$  via some  $h$ , the signed length of any signed path must be  $h(y) - h(x)$ .

Now for any  $i > \max\{c, |D_e|\}$ ,  $|u(i) - *| > i > |D_e|$  for any  $* \in U \setminus \{u(i)\}$ , and with  $|D_e \cap U^{\mathfrak{D}}| \geq 2$ , this gap is too big for there to exist a homomorphism sending  $d_e$  to  $u(i)$ . In other words,  $\mathfrak{D}_e \in \text{CSP}(\mathbb{Z}, \text{succ}, U) \iff \exists h \in \text{Hom}_{\text{succ}}(\mathfrak{D}_e, (\mathbb{Z}, \text{succ}, U))$  s.t.  $h(d_e) \in \{u(i)\}_{i \leq \max\{|D_e|, c\}}$ , which is what the algorithm checks.

As for the corollary claiming efficiency, note that the above has reduced  $\text{CSP}(\mathbb{Z}, \text{succ}, U)$  to  $O(|D|)$ -many tests of whether each  $h(d'_e) \in U$ . But this can be done by asking “is  $h(d'_e) = u(j)$ ?” for each  $j \in [0, \max\{c, |D_e|\} + 1]$ . Both  $h(d'_e)$  and each  $u(j)$  are of size polynomial in  $|D|$ , thanks to SR and the connectedness of  $\mathfrak{D}_e$ .

### A.2 Details of Section 3, Theorem 1 and Corollary 2

*Proof (Section 3, Theorem 1).* Again, thanks to EMLG one needs to shift a succ-homomorphism, if found, at most linearly many times, and SR bars exponential blow-ups.

<sup>10</sup> Note: if for some  $i$  both  $(x_i, x_{i+1})$  and  $(x_{i+1}, x_i)$  are in  $E$  then the graph cannot homomorphically map to  $(\mathbb{Z}, \text{succ})$ ; if between  $x, y$  there exist two different signed paths with different signed lengths, the graph is also not in  $\text{CSP}(\mathbb{Z}, \text{succ})$ .

Upon input  $\mathfrak{D} = (D, \text{succ}^{\mathfrak{D}}, U_1^{\mathfrak{D}}, \dots, U_k^{\mathfrak{D}})$ , for each component  $\mathfrak{D}_e$ :

1. If  $|D_e \cap (\bigcup_j U_j^{\mathfrak{D}})| \leq 1$ , like in Proposition 1 the gaps within and across  $U_j$ 's make no impact on existence of homomorphisms. One just needs to run the  $\text{CSP}(\mathbb{Z}, \text{succ})$  decider / searcher starting with assigning  $d_e$  to an arbitrary  $b \in U_j$ , where  $U_j^{\mathfrak{D}}$  is the unique unary intersecting  $D_e$  with  $d_e$  the unique intersection.
2. Otherwise, let  $c := \max$  of the  $\leq k + \binom{k}{2}$  constants witnessing EMLG. Then

for any  $(j, l) \in k^2$ , any  $n > \max\{c, |D_e|\}$ , any  $n' \in \begin{cases} \mathbb{N} & \text{if } l \neq j \\ \mathbb{N} \setminus \{n\} & \text{o.w.} \end{cases}$

we have  $|u(n, j) - u(n', l)| > n > |D_e|$ . It follows that one needs at most  $O(|D_e|) \sim O(|D|)$  iterations, for  $n \in [0, \max\{c, |D_e|\}]$ .

Fix arbitrary  $d_e \in D_e \cap (\bigcup_j U_j^{\mathfrak{D}})$  before iterating. Within each iteration, run  $\text{CSP}(\mathbb{Z}, \text{succ})$  assigning  $d_e$  to some  $u(n, y)$  where  $d_e \in D_e \cap U_y^{\mathfrak{D}}$ ,  $n$  is the iteration number ( $n \leq \max\{c, |D_e|\}$ ). Since  $U_y$  has SR,  $u(n, y)$  has size polynomial in  $n$ , and  $n$  is linear in  $|D|$ . If a succ-homomorphism  $h$  is found, verify whether  $h(d'_e) \in U_{y'}$  for each  $d'_e \in D_e \cap U_{y'}^{\mathfrak{D}}$ . Via  $U_{y'} \leq_m^p$  (hence  $\leq_T^p \varinjlim_j U_j$ ), the decision of  $U_{y'}$ -membership of  $h(d'_e)$  reduces to calls to the  $\varinjlim_j U_j$ -oracle. Note each  $h(d'_e) \in [u(n, y) - |D_e|, u(n, y) + |D_e|]$ , so the reduction is of time polynomial w.r.t.  $|D|$ .  $\square$

*Proof (Section 3, Corollary 2).* As in Corollary 1, for each  $h(d'_e)$ , ask if  $h(d'_e) = u(s, t)$  for each  $t \in [1, k]$  involved,  $s \in [0, \max\{c, |D_e|\} + 1]$ . By SR and connectedness, both  $h(d'_e)$  and each  $u(s, t)$  have size polynomial in  $|D|$ .  $\square$

### A.3 Details of Section 4, Proposition 2 and Corollary 3

*Proof (Section 4, Proposition 2).* Upon each input  $\mathfrak{D} = (D, \text{succ}^{\mathfrak{D}}, U^{\mathfrak{D}}, \mathbf{0}^{\mathfrak{D}})$ , let  $\mathfrak{D}_0 :=$  the unique component containing  $\mathbf{0}^{\mathfrak{D}}$ . Call  $\text{CSP}(\mathbb{Z}, \text{succ}, U)$ -oracle on  $\mathfrak{D} \setminus \mathfrak{D}_0$  and echo if rejects.

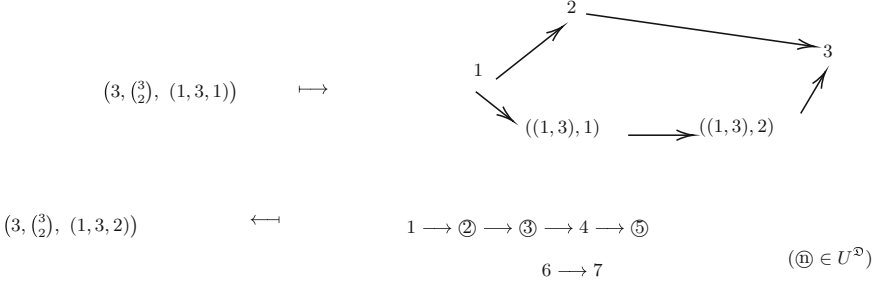
Call  $\text{CSP}(\mathbb{Z}, \text{succ})$ -solver on  $(D_0, \text{succ}^{\mathfrak{D}} \cap D_0^2)$  initiated by sending  $\mathbf{0}^{\mathfrak{D}}$  to 0, and echo if rejects; otherwise, the unique succ-homomorphism  $h : (D_0, \text{succ}^{\mathfrak{D}} \cap D_0^2) \rightarrow (\mathbb{Z}, \text{succ})$  that satisfies  $h(\mathbf{0}^{\mathfrak{D}}) = 0$  has been found, and one just needs to verify if for each  $d$  we have  $h(d) \in U$  by calling the  $U$ -oracle. Note each  $h(d) \in [0 - |D_0|, 0 + |D_0|]$  so input size remains polynomial in  $|D_0| \leq |D|$ .

The only oracles we called above are the  $U$ -oracles and  $\text{CSP}(\mathbb{Z}, \text{succ}, U)$ -oracles, both of which Karp (hence Cook) reduces to the  $\varinjlim$ -oracle on RHS.

*Proof (Section 4, Corollary 3).* If  $U \in \mathbb{P}$ , calls to the  $U$ -oracle counts as polynomial time overhead; if  $U \leq_T^p \text{CSP}(\mathbb{Z}, \text{succ}, U)$ , note the direct limit can be computed as  $\begin{cases} \chi_U(\frac{n-1}{2}) & \text{if } n = 2k + 1 \\ \chi_{\text{CSP}(\mathbb{Z}, \text{succ}, U)}(\frac{n}{2}) & \text{o.w.} \end{cases}$ , where  $\chi_S(\bullet)$  is the membership function of set  $S$ , and  $\chi_U$  can be computed by computing  $\chi_{\text{CSP}(\mathbb{Z}, \text{succ}, U)}$  now at polynomial cost.

**A.4 Details of Section 4, Theorem 2**

*Proof.* Before going into the full details, we illustrate the reductions with the following example (Fig. 1):



**Fig. 1.** Example of Theorem 2 Reduction

1.  $\leq_m^p$ : upon  $(n, S_1, S_2)$ , construct  $D := \{1, \dots, n\}$ ,  $\text{succ}^{\mathfrak{D}} = \emptyset$ ,  $U^{\mathfrak{D}} := D$ . Then for each  $(i < j) \in S_1$ :

- (a) If  $S_2(i, j) > 0$ , add intermediate vertices and edges  $i = ((i, j), 0) \xrightarrow{\text{succ}^{\mathfrak{D}}} ((i, j), 1) \xrightarrow{\text{succ}^{\mathfrak{D}}} \dots \xrightarrow{\text{succ}^{\mathfrak{D}}} ((i, j), S_2(i, j)) = j$ .
- (b) If  $S_2(i, j) < 0$ , add intermediate vertices and edges  $i = ((i, j), 0) \xleftarrow{\text{succ}^{\mathfrak{D}}} ((i, j), 1) \xleftarrow{\text{succ}^{\mathfrak{D}}} \dots \xleftarrow{\text{succ}^{\mathfrak{D}}} ((i, j), |S_2(i, j)|) = j$ .
- (c) if  $S_2(i, j) = 0$ , collapse vertices  $i \sim j$  (so now the “vertex classes”  $[i] = [j]$ ).

Now  $D = \{[1], \dots, [n]\} \cup \{((i, j), m)\}_{S_2(i,j) \neq 0, m \in [1, |S_2(i,j)|-1]}$  and  $U^{\mathfrak{D}} = \{[1], \dots, [n]\}$ . Note  $|D| \leq n + n \times \binom{n}{2}$ , and as  $S_1$  is part of the tuple,  $|D|$  is polynomial in  $|\langle (n, S_1, S_2) \rangle|$ , so  $(n, S_1, S_2) \mapsto \mathfrak{D} = (D, \text{succ}^{\mathfrak{D}}, U^{\mathfrak{D}})$  is a polynomial-time construction. Further,

- (a) If  $(n, S_1, S_2) \in \text{Gap}(U)$  witnessed by  $t \in U^n$ , let  $h : D \rightarrow \mathbb{Z}$ ,  $[j] \mapsto t(j)$  ( $j \in [1, n]$ ) and extend naturally to the intermediates, i.e.  $((i, j), m) \mapsto t(i) + \text{sign}(S_2(i, j)) \times m$ . The map puts  $\mathfrak{D} \in \text{CSP}(\mathbb{Z}, \text{succ}, U)$ .
- (b) If  $\mathfrak{D} \in \text{CSP}(\mathbb{Z}, \text{succ}, U)$  by  $h$ , then  $(t(i) = h([i]))_{i \in [1, n]}$  witnesses  $(n, S_1, S_2) \in \text{Gap}(U)$ .

2.  $\geq_m^p$ : Observe that  $s_1 := \left(3, \binom{3}{2}, \underbrace{\left(\binom{1}{1,2}, \binom{3}{1,3}, \binom{1}{2,3}\right)}\right) \notin \text{Gap}(U)$  due to

triangle-inequality, and taking arbitrary  $u_1 \neq u_2 \in U$  (not  $U^{\mathfrak{D}}$ ),  $s_2 := \left(2, \left\{\binom{1}{1,2}\right\}, (u_2 - u_1)\right) \in \text{Gap}(U)$ .  $s_1, s_2$  are input-independent.

Now let  $\mathfrak{D} = (D, \text{succ}^{\mathfrak{D}}, U^{\mathfrak{D}})$ . We want to produce a short tuple  $s_{\mathfrak{D}} = (n, S_1, S_2)$  in time polynomial in  $|D|$ . We present an algorithm for that.

- (a) Run  $\text{CSP}(\mathbb{Z}, \text{succ})$  decider/searcher on  $\mathfrak{D}$ , and halt setting  $s_{\mathfrak{D}} := s_1$  if **reject** was returned; else, we obtain an  $h \in \text{Hom}_{\text{succ}}((D, \text{succ}^{\mathfrak{D}}), (\mathbb{Z}, \text{succ}))$  from the run.
- (b) Throw away the components  $\mathfrak{D}_e$  with  $|D_e \cap U^{\mathfrak{D}}| \leq 1$ , keeping  $\mathfrak{D}' := \prod_{e: |D_e \cap U^{\mathfrak{D}}| \geq 2} \mathfrak{D}_e$ . If  $\mathfrak{D}' = \emptyset$ , halt setting  $s_{\mathfrak{D}} := s_2$ .
- (c) Let  $n := |U^{\mathfrak{D}} \cap D'|$ ; now  $n \in [2, |D'|]$ . Enumerate elements in  $U^{\mathfrak{D}} \cap D'$  as  $v_1, \dots, v_n$ ; let  $S_1 := \{(i < j) : v_i, v_j \text{ are from the same component}\}$ . Let  $S_2(i, j) := h(v_j) - h(v_i)$  with  $h$  from Step 2a. Let  $s_{\mathfrak{D}} := \langle (n, S_1, S_2) \rangle$ .  $s_{\mathfrak{D}}$  has size polynomial in  $|D|$  and the algorithm is efficient. Correctness:
  - (a) Assume  $\mathfrak{D} \in \text{CSP}(\mathbb{Z}, \text{succ}, U)$  witnessed by  $h'$ , then  $s_{\mathfrak{D}} \neq s_1$  for  $h' \in \text{Hom}_{\text{succ}}((D, \text{succ}^{\mathfrak{D}}), (\mathbb{Z}, \text{succ}))$  and the  $\text{CSP}(\mathbb{Z}, \text{succ})$ -decider should've caught a homomorphism.
    - If  $\mathfrak{D}' \neq \emptyset$ : for each  $v_i \neq v_j$  from the same component in  $\mathfrak{D}'$ ,  $h'(v_j) - h'(v_i) = h(v_j) - h(v_i)$ . Therefore setting  $t_i = h'(v_i)$  for each  $i \in [1, n]$ , we get a witness for  $(n, S_1, S_2) \in \text{Gap}(U)$ .
    - If  $\mathfrak{D}' = \emptyset$ , we exited with an  $s_2 \in \text{Gap}(U)$ .
  - (b) Assume  $s_{\mathfrak{D}} \in \text{Gap}(U)$ , then again  $s_{\mathfrak{D}} \neq s_1$ .
    - If  $s_{\mathfrak{D}} = s_2$  then the  $h$  from Step 2a translates componentwise and then patches to a  $\{\text{succ}, U\}$ -homomorphism.
    - otherwise there exists  $t \in U^n$  with  $t(j) - t(i) = S_2(i, j) \ (\forall (i < j) \in S_1)$ . Each component in  $\mathfrak{D}'$  contains a  $v_{i_e} \in U^{\mathfrak{D}} \cap D'$ ; shift  $h \upharpoonright_{\mathfrak{D}_e}$  s.t.  $h(v_{i_e}) = t(i_e)$ . Now for each  $v_{j_e} \neq v_{i_e} \in D_e \cap U^{\mathfrak{D}}$ ,  $h(v_{j_e}) = h(v_{i_e}) + S_2(i_e, j_e) = t(j_e) \in U$ . Do so for each component of  $\mathfrak{D}'$ . Lastly for each component in  $\mathfrak{D} \setminus \mathfrak{D}'$ , shift  $h$  if need be. □

### A.5 Details of Section 5, Fact 3 and Proposition 4

We first prove the folklore / exercise in Sect. 5, Fact 3, conceptually by mimicing the Search to Decision reduction for SAT. As in graphs, a *core*  $\mathfrak{B} \subset_{\text{ind}} \mathfrak{A}$  is a minimal image of endomorphisms, i.e. there exists no  $\sigma \in \text{End}(\mathfrak{A})$  s.t.  $\sigma(\mathfrak{A}) \subsetneq \mathfrak{B}$ . For finite  $\mathfrak{A}$ 's, cores always exist and are unique up to isomorphism [3]. are isomorphic,<sup>11</sup> so hereinafter we talk about “the” core of  $\mathfrak{A}$ .

Let  $\mathfrak{B}$  be the core of  $\mathfrak{A}$ , then inclusion  $\mathfrak{B} \hookrightarrow \mathfrak{A}$  and the definitional endomorphism  $\mathfrak{A} \rightarrow \mathfrak{B}$  shows homomorphic equivalence of  $\mathfrak{A}$  and  $\mathfrak{B}$ , so it suffices to show  $\text{SEARCH-CSP}(\mathfrak{A}) \leq_T^p \text{CSP}(\mathfrak{B})$ .<sup>12</sup>

To make the description less cumbersome let's define POINTED-CSP: fix arbitrary  $b \in B$ . POINTED-CSP( $\mathfrak{B}, b$ ) takes a pair  $(\mathfrak{D}, d)$  with  $\mathfrak{D}$  a finite  $\tau$ -structure,  $d \in D$ , and returns 1 iff there exists  $h \in \text{Hom}_{\tau}(\mathfrak{D}, \mathfrak{B})$  s.t.  $h(d) = b$ .

*Claim:* POINTED-CSP( $\mathfrak{B}, b$ )  $\leq_m^p$  (hence  $\leq_T^p$ ) CSP( $\mathfrak{B}$ ).

<sup>11</sup> (e.g. [18] for proof of isomorphism of cores in the special case of graphs) Let  $\mathfrak{B}_1, \mathfrak{B}_2 \subset_{\text{ind}} \mathfrak{A}$  be cores, witnessed by  $e_1 : \mathfrak{A} \rightarrow \mathfrak{B}_1, e_2 : \mathfrak{A} \rightarrow \mathfrak{B}_2$ , two surjective homomorphisms (otherwise  $\mathfrak{B}_*$  is not a core). Then  $e_1|_{\mathfrak{B}_2}$  is also surjective, for otherwise  $(e_1|_{\mathfrak{B}_2}) \circ e_2$  gives an induced substructure of  $\mathfrak{B}_1$  which is an endomorphic image. Dually  $e_2|_{\mathfrak{B}_1}$  is surjective. Therefore  $|B_1| = |B_2|$ ,  $e_1|_{\mathfrak{B}_2}$  and  $e_2|_{\mathfrak{B}_1}$  are bijective homomorphisms, and we have the isomorphism.

<sup>12</sup> Note:  $\mathfrak{A}$  is input-independent, hence  $\text{core}(\mathfrak{A}) = \mathfrak{B}$ .

*Proof (Proof of Claim).* Upon a POINTED-CSP $(\mathfrak{B}, b)$  input  $(\mathfrak{D}, d)$ , let  $\tilde{\mathfrak{D}} := \frac{\mathfrak{D} \cup \mathfrak{B}}{b \sim d}$ .<sup>13</sup> This is a polynomial time construction since  $|B| \sim O(1)$ . I claim  $(\mathfrak{D}, d) \in \text{POINTED-CSP}(\mathfrak{B}, b) \iff \tilde{\mathfrak{D}} \in \text{CSP}(\mathfrak{B})$ .

1.  $\implies$  : let  $h : (\mathfrak{D}, d) \rightarrow (\mathfrak{B}, b)$  be a basepoint-preserving homomorphism. Then  $\tilde{h} : \tilde{\mathfrak{D}} \rightarrow \mathfrak{B}, h(x) := \begin{cases} h(x), & \text{if } x \in D \\ x, & \text{o.w.} \end{cases}$  is a homomorphism.
2.  $\impliedby$  : let  $h : \tilde{\mathfrak{D}} \rightarrow \mathfrak{B}$  be a homomorphism. Then  $h|_{\mathfrak{B}} : \mathfrak{B} \rightarrow \mathfrak{B}$  is an endomorphism of the finite core  $\mathfrak{B}$ . Since an endomorphism of a core is an automorphism,<sup>14</sup> there exists  $\sigma \in \text{Aut}(\mathfrak{B}) = \text{End}(\mathfrak{B})$  s.t.  $\sigma \circ (h|_{\mathfrak{B}}) = 1_{\mathfrak{B}}$ . Now  $\sigma \circ (h|_{\mathfrak{D}})$  is a homomorphism  $\mathfrak{D} \rightarrow \mathfrak{B}$  sending  $d$  to  $\sigma(h([d])) = \sigma(h([b])) = b$ , which is the desired witness.

Analogously if we define a  $m$ -POINTED-CSP $(\mathfrak{B}, b_1, \dots, b_m)$  it also Karp-reduces to CSP $(\mathfrak{B})$ , the reduction being  $(\mathfrak{D}, d_1, \dots, d_m) \rightsquigarrow \tilde{\mathfrak{D}} := \frac{\mathfrak{D} \cup \mathfrak{B}}{\{d_i \sim b_i\}_{i \in [1, m]}}$ .<sup>15</sup>

An algorithm that calls the CSP $(\mathfrak{B})$ -oracle to solve SEARCH-CSP $(\mathfrak{A})$  therefore goes as:

```

if  $\mathfrak{D} \notin \text{CSP}(\mathfrak{B})$  then
  | Reject

for  $i \in [1, |D|]$  do
  | for  $j \in [1, |B|]$  do
  | | if  $(\mathfrak{D}, d_1, \dots, d_i) \in \text{POINTED-CSP}(\mathfrak{B}, h(d_1), \dots, h(d_{i-1}), b_j)$  then
  | | | Set  $h(d_i) := b_j$ 
  | | | break //break inner iteration, go to next  $i$ 
  | | end
  | | reject
  | end
Algorithm 2: Querying CSP $(\mathfrak{B})$ -oracle to solve SEARCH-CSP $(\mathfrak{A})$ 

```

Algorithm 2 finds a full homomorphism  $\mathfrak{D} \rightarrow \mathfrak{B} \iff \mathfrak{D} \in \text{CSP}(\mathfrak{B}) = \text{CSP}(\mathfrak{A})$  so it solves SEARCH-CSP $(\mathfrak{A})$ ; furthermore it calls the CSP $(\mathfrak{B})$ -oracle  $O(nk)$  times.  $\implies$  is by definition, and to see  $\impliedby$ , note that the existence

<sup>13</sup> That is, take a copy of  $\mathfrak{D}$  and a copy of  $\mathfrak{B}$ , glue  $b$  and  $d$ .

<sup>14</sup>  $e : \mathfrak{B} \rightarrow \mathfrak{B}$  must be surjective, so bijective. As  $\text{Im} e \subseteq_{\text{ind}} \mathfrak{B}$ , the bijectivity says  $\text{Im} e = \mathfrak{B}$ . In particular this says  $e$  preserves the number of solutions for each  $R_i$ , i.e. there cannot be  $(a_1, \dots, a_{r_i}) \notin R_i^{\mathfrak{B}}$  s.t.  $(e(a_1), \dots, e(a_{r_i})) \in R_i^{\mathfrak{B}}$ . It follows that  $e$  preserves the relations bidirectionally, hence an automorphism.

<sup>15</sup> Note e.g. when  $m = 2$  in the  $\impliedby$  direction has  $\sigma \circ h|_{\mathfrak{D}}$  sending  $[d_i]$  to  $[b_i]$  for both  $i$  as each  $b_i \in B$ .

of an  $h : \mathfrak{D} \rightarrow \mathfrak{B}$  means for each  $i \in [1, |D|]$  there exists some  $j \in [1, |B|]$  s.t.  $h(d_i) = b_j$ , i.e. the existence of an accepting path.<sup>16</sup>  $\square$

Next,

*Proof (Proposition 4).*

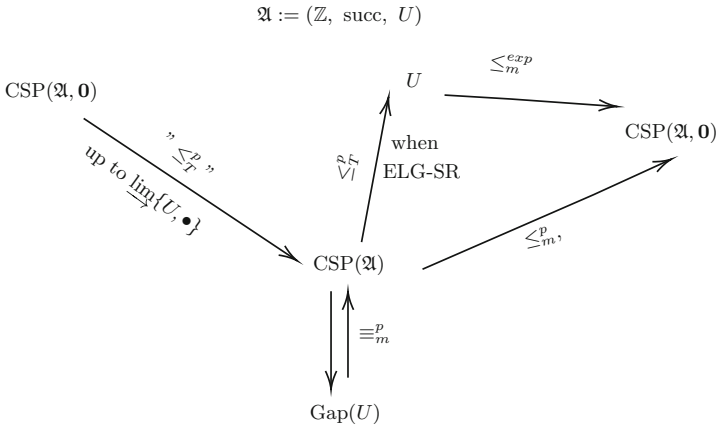
As before, decompose  $\mathfrak{D} = (D, E^{\mathfrak{D}}, U^{\mathfrak{D}}, \mathbf{0}^{\mathfrak{D}})$  into connected components and let the  $\text{CSP}(\mathfrak{A})$  handle the non central components.

For the central component, call  $\text{SEARCH-CSP}(\mathfrak{A}^\downarrow[A^\downarrow_0])$  oracle, which thanks to  $|A^\downarrow_0| < \aleph_0$  and Fact 3 Cook reduces to  $\text{CSP}(\mathfrak{A}^\downarrow[A^\downarrow_0])$ . If we found an  $h \in \text{Hom}_E((D, E^{\mathfrak{D}}), \mathfrak{A}^\downarrow[A^\downarrow_0])$ , by the assumption, all  $h' \in \text{Hom}_E((D, E^{\mathfrak{D}}), \mathfrak{A}^\downarrow[A^\downarrow_0])$  is of the form  $\sigma h$  for different  $\sigma$ 's in  $\text{End}_E(\mathfrak{A}^\downarrow[A^\downarrow_0])$ .

Note that  $|A^\downarrow_0| < \aleph_0 \implies |\text{End}_E(\mathfrak{A}^\downarrow[A^\downarrow_0])| \leq |A^\downarrow_0|^{|A^\downarrow_0|} < \aleph_0$  and  $|\text{End}_E(\mathfrak{A}^\downarrow[A^\downarrow_0])|$  is input-independent. Each  $\sigma_i \in |\text{End}_E(\mathfrak{A}^\downarrow[A^\downarrow_0])|$  is a finite domain function, whose encoding is also input-independent. For each of the  $O(1)$ -many  $\sigma_i \in |\text{End}_E(\mathfrak{A}^\downarrow[A^\downarrow_0])|$ , compute  $\sigma_i h$  (Since  $\sigma_i$  is input-independent and  $A^\downarrow_0$  is constant-sized, time and space cost is dominated by computing  $h$ , which is again dominated by the oracle call to  $\text{CSP}(\mathfrak{A}^\downarrow[A^\downarrow_0])$ ); then check if  $\sigma_i h$  preserves  $\mathbf{0}^{\mathfrak{D}}$  in  $O(1)$  time and if  $\sigma_i h$  preserves the unary by calling the  $U$ -oracle.  $\square$

## B Diagrams Summarizing Main Results

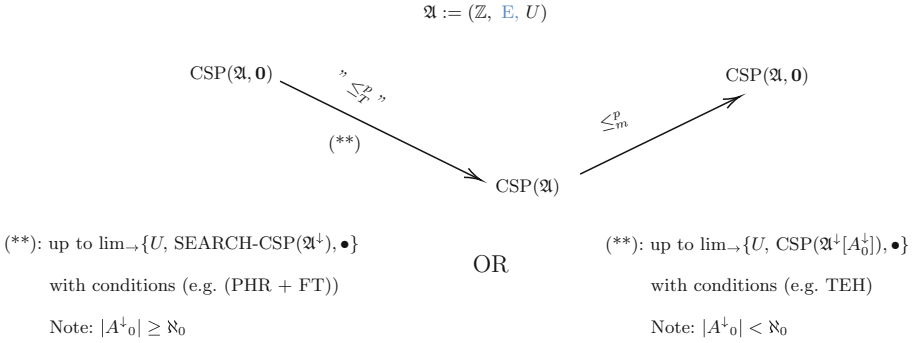
### B.1 For Section 4



**Fig. 2.** Bounds and Equivalences of  $\text{CSP}(\mathbb{Z}, \text{succ}, U)$

<sup>16</sup> The algorithm also has the benefit that whenever a homomorphism is found, the solution tuple  $(h(d_1), \dots, h(d_{|D|})) = (b_{s,1}, \dots, b_{s,|D|})$  is the minimal element in the dictionary order by indices in the target. That is, for any other solution  $(b'_{s,1}, \dots, b'_{s,|D|}) \in B^{|D|} = (b_1 \prec \dots \prec b_{|B|})^{|D|}$ ,  $(b_{s,1}, \dots, b_{s,|D|}) \preceq (b'_{s,1}, \dots, b'_{s,|D|})$ .

### B.2 For Section 5



**Fig. 3.** Results for  $\mathfrak{A} := (\mathbb{Z}, E, U), \mathfrak{A}^\perp := (\mathbb{Z}, E)$ . Recall  $\mathfrak{A}^\perp_0$  is the  $\mathbf{0}$ -component of  $(\mathfrak{A}^\perp, \mathbf{0})$ , and  $A^\perp_0$  its domain.

## C Detailed Verification of Miscellaneous Claims

### C.1 Full Details of Section 3, Observation 1

We give more details on the polynomial-time algorithm. Modify the greedy algorithm that determines if an input  $G \in \text{CSP}(\mathbb{Z}, \text{succ})$ . For each component  $\mathfrak{D}_e \subseteq \mathfrak{D} = (D, \text{succ}^\mathfrak{D}, U^\mathfrak{D})$ , if  $D_e \cap U^\mathfrak{D} = \emptyset$ , run the  $\text{CSP}(\mathbb{Z}, \text{succ})$  decider on  $\mathfrak{D}_e$ ; otherwise pick an arbitrary  $d_e \in U^\mathfrak{D} \cap D_e$ , then run the greedy  $\text{CSP}(\mathbb{Z}, \text{succ})$  algorithm on  $\mathfrak{D}_e$  initiated by mapping  $d_e$  to  $b$ . When the algorithm accepts, it yields a  $\{\text{succ}\}$ -homomorphism  $h_e : (D, \text{succ}^\mathfrak{D}) \rightarrow (\mathbb{Z}, \text{succ})$ ; accepts on this component only if  $h_e(d'_e) \in U$  for each  $d'_e \in U^\mathfrak{D} \cap D_e$ . Accept  $\mathfrak{D}$  only if all components accept.

If the algorithm above accepts, obviously each  $h_e$  is a  $\{\text{succ}, U\}$ -homomorphism; conversely if  $\mathfrak{D} \in \text{CSP}(\mathbb{Z}, \text{succ}, U)$ , so is each component  $\mathfrak{D}_e$ . Hence there exists some homomorphism  $h_e$  sending  $d_e$  to some  $x \in b + a\mathbb{Z}$ . Now  $\text{Aut}(\mathbb{Z}, \text{succ}, U) \cong \mathbb{Z}$  composes of translations by  $ak$  ( $k \in \mathbb{Z}$ ), so there exists  $h'_e \in \text{Hom}_{\text{succ}, U}(\mathfrak{D}_e, (\mathbb{Z}, \text{succ}, U))$  sending  $d_e$  to  $b$ , which is the one affording acceptance of  $\mathfrak{D}_e$ .

### C.2 Section 3 Equation 1 Gives the L.U.B. in Karp-Order

To see that  $\varinjlim_j U_j$  defined as above is indeed the least upper bound, observe that each  $U_r \leq_m^p \varinjlim_j U_j$  via  $m \mapsto mk + (r \bmod k)$ , and if  $Q \subseteq \mathbb{Z}$  is s.t.  $U_r \leq_m^p Q$  via  $f_r$  for each  $r \in [k]$ , then  $\varinjlim_j U_j \leq_m^p Q$  via  $x \mapsto f_r(\frac{x - (x \bmod k)}{k})$  where

$$r = \begin{cases} x \bmod k & \text{if } k \nmid x \\ k & \text{o.w.} \end{cases}$$

### C.3 More Discussion on Diff<sub>d</sub>

*Proof (details of Section 5, Observation 2).*

1. For each component  $\mathfrak{D}_e$  s.t.  $\mathbf{0}^{\mathfrak{D}} \notin \mathfrak{D}_e$ , call the CSP( $\mathbb{Z}, \text{succ}, U$ )-oracle, which reduces to the  $\varinjlim \{U, \text{CSP}(\mathbb{Z}, \text{Diff}_d, U)\}$ -oracle.
2. For the unique component  $\mathfrak{D}_0 \ni \mathbf{0}^{\mathfrak{D}}$ , run the greedy, efficient SEARCH-CSP( $\mathbb{Z}, \text{Diff}_d$ ) algorithm to decide if there is a (unique)  $\{\text{Diff}_d, \mathbf{0}\}$ -homomorphism  $h$ . **Reject** if none found. Having found  $h$ , for each  $u \in U^{\mathfrak{D}} \cap D_0$ , call  $U$ -oracle on  $h(u)$ . Note that  $h(u) \in [-|D|, |D|]$  and the oracle call once again reduces to calling  $\varinjlim \{U, \text{CSP}(\mathbb{Z}, \text{Diff}_d, U)\}$ .

Other interesting properties of CSP( $\mathbb{Z}, \text{Diff}_d, U$ ) may be said. For example:

1. One may define *eventually (mutually) d-largely gapped* ( $E(M)LG_d$ ) analogously to Definitions 1, 3, changing “ $|u(n, *) - u(n', *)| > n$ ” into “ $|u(n, *) - u(n', *)| > dn$ ”. Keeping the definition of SR (2) unchanged, we have CSP( $\mathbb{Z}, \text{Diff}_d, U$ )  $\leq_T^p U$  when ELG<sub>d</sub>-SR, and CSP( $\mathbb{Z}, \text{Diff}_d, U$ )  $\leq_T^p \varinjlim U$  when EMLG<sub>d</sub>-SR.
2. Note that  $\text{Aut}(\mathbb{Z}, \text{Diff}_d) \cong S_d \times (d\mathbb{Z})^d \cong S_d \times \mathbb{Z}^d$  as groups: indeed, an automorphism permutes the components and then shifts on each component by a distance in  $d\mathbb{Z}$ .

### C.4 Figure 4 Disproves Converse of Proposition 3(1)

We show that the converse of Proposition 3(1) is not true; in fact, even  $(\text{End}_{E, \mathbf{0}}(\mathfrak{A}^{\perp_0}) = \{\mathbf{1}\}) + \text{FT} \not\Rightarrow \text{PHR}$ .<sup>17</sup> We claim that Fig. 4 below has  $(\text{End}_{E, \mathbf{0}}(\mathfrak{A}^{\perp_0}) = \{\mathbf{1}\}) + \text{FT}$  but no PHR:  $\mathfrak{D}$  embeds in  $(\mathfrak{A}^{\perp}, \mathbf{0})$  in 2 ways.

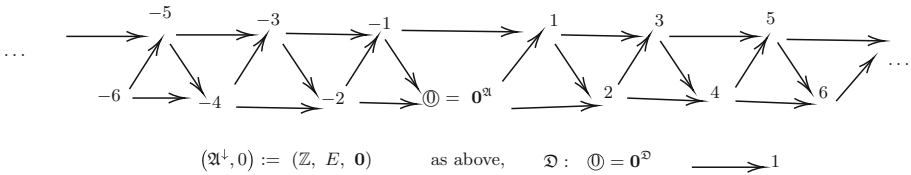


Fig. 4. Counter example  $\mathfrak{A}^{\perp}$

We verify that Fig. 4 has  $(\text{End}_{E, \mathbf{0}}(\mathfrak{A}^{\perp_0}) = \{\mathbf{1}\}) + \text{FT}$  yet no PHR. Without naming the constant, each vertex has in-degree 2 and out-degree 2, so any

<sup>17</sup> If the implication were true, for Theorem 3 we would have algebraic conditions without reference of “all finite connected  $\tau$ -structures”, which would be considerably cleaner.



translation is an  $\{E\}$ -endomorphism — in fact even an automorphism, yielding FT. Furthermore, after naming  $\mathbf{0}^{\mathfrak{A}} := 0$ , its predecessors and successors are fixed: let  $\sigma \in \text{End}_{E, \mathbf{0}}(\mathfrak{A}^\downarrow, \mathbf{0})$ .  $\sigma(2), \sigma(1) \in \{2, 1\}$  yet if  $\sigma(2) = 1$ , neither  $\sigma(1) = 1$  nor  $\sigma(1) = 2$  would give  $E^{\mathfrak{A}}(\sigma(1), \sigma(2))$ . It follows that  $\sigma(2) = 2$ . Likewise,  $\sigma(\pm 1) = \pm 1$  and  $\sigma(-2) = -2$ . For  $|n| \geq 3$ ,  $\sigma(n) = n$  follows from induction. The fact that vertices further away are fixed follow from induction, so  $\text{End}_{E, \mathbf{0}}(\mathfrak{A}^\downarrow_0) = \text{End}_{E, \mathbf{0}}(\mathfrak{A}^\downarrow, 0) = \{1\}$ . On the other hand,  $\mathfrak{D} := \mathbf{0}^{\mathfrak{D}} = \mathbf{0} \rightarrow 1$  embeds into both  $\mathbf{0} \rightarrow 1 \subset \mathfrak{A}$  and  $\mathbf{0} \rightarrow 2 \subset \mathfrak{A}$ .

**Remark 1.** *One might also want to note that a claim stronger than Prop. 3(1), that  $\text{PHR} \implies \text{End}_{E, \mathbf{0}}(\mathfrak{A}^\downarrow, \mathbf{0}) = \{1\}$ , is also false. Consider  $\mathfrak{A}^\downarrow := (\mathbb{Z}; E, \mathbf{0} := 0)$  with edges defined as*

$$\dots \rightarrow -8 \rightarrow -7 \rightarrow -6 \rightarrow -5 \quad -4 \leftarrow -3 \quad -2 \leftarrow -1 \quad \mathbf{0} \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow \dots$$

*Any connected finite  $\{E, \mathbf{0}\}$ -structure must homomorphically map to the non-negative, so a  $\{E, \mathbf{0}\}$ -homomorphism exists iff  $\mathfrak{D}$  is loopless, directed acyclic, and each vertex has a non-negative signed distance from  $\mathbf{0}^{\mathfrak{D}}$  (i.e. never “ $a \rightarrow \mathbf{0}^{\mathfrak{D}}$ ”). Each  $\{E, \mathbf{0}\}$ -homomorphism from such  $\mathfrak{D}$ 's should it exist, must be unique, mapping each vertex to its distance from  $\mathbf{0}^{\mathfrak{D}}$ . However  $\text{End}_{\{E, \mathbf{0}\}}(\mathfrak{A}^\downarrow, \mathbf{0})$  contains swapping  $-4 \leftarrow -3$  with  $-2 \leftarrow -1$ .*

### C.5 Definition of Connectedness in General Case (Re. Comments at Beginning of Section 5)

One may even have freedoms on the arities of the non unaries  $\{E_j\}_j$ . In these cases, call  $x, y \in \mathbb{Z}$  “connected” if

1. (Base) there exists some  $j \in [1, k]$  s.t.  $(\dots, x, \dots, y, \dots)$  or  $(\dots, y, \dots, x, \dots) \in E_j^{\mathfrak{D}}$ . OR
2.  $\exists z$  s.t.  $x, z$  are connected and  $z, y$  are connected.

Call  $\mathfrak{D}' \subset_{\text{ind}} \mathfrak{D}$  a connected component of  $\mathfrak{D}$  if for any  $x \neq y \in \mathfrak{D}'$ ,  $x$  and  $y$  are connected; and  $\forall x \in \mathfrak{D}', \forall z \notin \mathfrak{D}', x, z$  are not connected. (This happens if and only if the Gaifman graph of  $\mathfrak{D}'$  is a connected component of the Gaifman graph of  $\mathfrak{D}$ .) Vacuously, an isolated vertex (an  $x \in D$  that does not appear in any  $E_j^{\mathfrak{D}}$  with arity  $\geq 2$ ) is a connected component.

A  $\{E_j, \mathbf{c}\}_{j \in [1, k]}$ -structure  $\mathfrak{D}$  is connected if  $\mathfrak{D}$  itself is a connected component; equivalently,  $\mathfrak{D}$  is not the disjoint union of two or more connected components.

A notable property relevant to later discussion: when a finite, connected  $\{E_j\}_{j \in [k]}$ -structure  $\mathfrak{D}$  has  $(E_j^{\mathfrak{D}} = \emptyset)_{j \in [k]}$  then  $|\text{Hom}_{\{E_j\}_{j \in [k]}, \mathbf{c}}(\mathfrak{D}, (\mathbb{Z}, (E_j^{\mathfrak{A}})_j, \mathbf{c}^{\mathfrak{A}}))| = 1$  for free, because  $D = \{\mathbf{c}^{\mathfrak{D}}\}$  in this case:  $\mathbf{c}^{\mathfrak{D}}$  is its own connected component and  $\mathfrak{D}$  is connected.

### C.6 Verification of Corollary 4

*Proof.* With assumption set 1, by  $U \in \text{P}$  and  $\text{CSP}(\mathfrak{A}^\downarrow) \leq_m^{\text{P}} \text{CSP}(\mathfrak{A})$ , all other terms in the direct limit are reduced to  $\text{CSP}(\mathfrak{A})$ . Likewise for assumption set 2.

### C.7 Remark on Further Generalizing Sect. 5

1. The converses of Proposition 2-type reductions, i.e.  $(\mathbb{Z}, E, U) \leq_m^p (\mathbb{Z}, E, U, \mathbf{0})$ , always hold by the identity reduction (as pp-sentence).<sup>18</sup>
2. All arguments in Theorem 3 to Proposition 4 generalize to  $(\mathbb{Z}, \mathbf{E}, \mathbf{U})$ , i.e. with finitely many binaries and finitely many unaries;  $\mathbf{0}$  may be changed to  $\mathbf{c}$  for any  $\mathbf{c} \in \mathbb{Z}$ . In these general cases we need a careful definition of connected components, and the arguments would be longer, but still completely analogous. For the ease of presentation, we assume our current presentation.

### C.8 More Discussion on Future Directions

Section 1, we believe these are solid first steps towards characterizing the CSP of non- $\omega$ -categorical binary structures. In fact, one meaningful subclass of non- $\omega$ -categorical structures where  $(\mathbb{Z}, \text{succ}, U)$  lives is those that are *mutually algebraic* (e.g. [12, 21]) ones, which also include all  $\mathfrak{A} = (\mathbb{Z}, E, U)$ 's where the underlying  $\mathfrak{A}^\downarrow$  has (*universally*) *bounded degree*. Hence one may ask:

**Problem 4.** Let  $\mathfrak{A}^\downarrow = (\mathbb{Z}, E)$  be of degree  $d \geq 2$ , i.e.  $\text{deg}_{\mathfrak{A}^\downarrow}(v) \leq d$  for any  $v \in \mathbb{Z}$ .

1. Is  $\text{CSP}(\mathfrak{A}^\downarrow)$  either in P or in NPC? If not, are there meaningful dividing lines for tractability at least?
2. Can we fully characterize the complexity of  $\text{CSP}(\mathbb{Z}, E, U)$  ( $U \subseteq \mathbb{Z}$ ) now that we further restricted the underlying  $E$  by degree?

Some facts to bear in mind: that in general it is not true that the CSP of infinite digraphs have a P-NPC dichotomy [3, 4].<sup>19</sup> On the other hand, for any  $d \geq 2$  the disjoint union of all (isomorphic types of) finite *undirected* graph of degree  $\leq 2$  has the same CSP as  $K_{d+1}$ , by e.g. Brook's theorem.

## References

1. Barto, L., Kozik, M., Niven, T.: The CSP dichotomy holds for digraphs with no sources and no sinks (a positive answer to a conjecture of bang-jensen and hell). *SIAM J. Comput.* **38**(5), 1782–1802 (2009)
2. Bodirsky, M.: Graph homomorphisms and universal algebra course notes. TU Dresden (2015)
3. Bodirsky, M.: Complexity of Infinite-Domain Constraint Satisfaction, vol. 52. Cambridge University Press (2021)
4. Bodirsky, M., Grohe, M.: Non-dichotomies in constraint satisfaction complexity. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfssdóttir, A., Walukiewicz, I. (eds.) *ICALP 2008*. LNCS, vol. 5126, pp. 184–196. Springer, Heidelberg (2008). [https://doi.org/10.1007/978-3-540-70583-3\\_16](https://doi.org/10.1007/978-3-540-70583-3_16)

<sup>18</sup> Similarly by  $(\mathbb{Z}, E) \leq_m^p (\mathbb{Z}, E, U)$ , we know there are  $(\mathbb{Z}, E, U)$ 's whose CSP is not in NP: just let the underlying  $(\mathbb{Z}, E)$  be such that its CSP is CoNP-complete, which exists by Theorem 13.3.1 [3].

<sup>19</sup> See also, statuses of some other open questions in [3] or [here](#).

5. Bodirsky, M., Kára, J.: The complexity of temporal constraint satisfaction problems. *J. ACM (JACM)* **57**(2), 1–41 (2010)
6. Bodirsky, M., Mamino, M.: Constraint satisfaction problems over numeric domains. In: Dagstuhl Follow-Ups, vol. 7, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2017)
7. Bodirsky, M., Martin, B., Mamino, M., Mottet, A.: The complexity of disjunctive linear diophantine constraints. arXiv preprint [arXiv:1807.00985](https://arxiv.org/abs/1807.00985) (2018)
8. Bodirsky, M., Martin, B., Mottet, A.: Constraint satisfaction problems over the integers with successor. In: Halldórsson, M.M., Iwama, K., Kobayashi, N., Speckmann, B. (eds.) *ICALP 2015. LNCS*, vol. 9134, pp. 256–267. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-47672-7\\_21](https://doi.org/10.1007/978-3-662-47672-7_21)
9. Bodirsky, M., Martin, B., Mottet, A.: Discrete temporal constraint satisfaction problems. *J. ACM (JACM)* **65**(2), 1–41 (2018)
10. Bodirsky, M., Martin, B., Pinsker, M., Pongrácz, A.: Constraint satisfaction problems for reducts of homogeneous graphs. *SIAM J. Comput.* **48**(4), 1224–1264 (2019)
11. Bodirsky, M., Pinsker, M.: Schaefer's theorem for graphs. *J. ACM (JACM)* **62**(3), 1–52 (2015)
12. Braunfeld, S., Laskowski, M.C.: Mutual algebraicity and cellularity. arXiv preprint [arXiv:1911.06303](https://arxiv.org/abs/1911.06303) (2019)
13. Bulatov, A.A.: A dichotomy theorem for constraints on a three-element set. In: *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings*, pp. 649–658. IEEE (2002)
14. Bulatov, A.A.: A dichotomy theorem for nonuniform CSPs. In: *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 319–330. IEEE (2017)
15. Chen, H.: The complexity of quantified constraint satisfaction: collapsibility, sink algebras, and the three-element case. *SIAM J. Comput.* **37**(5), 1674–1701 (2008)
16. Feder, T., Vardi, M.Y.: The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory. *SIAM J. Comput.* **28**(1), 57–104 (1998)
17. Hell, P., Nešetřil, J.: On the complexity of h-coloring. *J. Comb. Theor. Ser. B* **48**(1), 92–110 (1990)
18. Hell, P., Nešetřil, J.: The core of a graph. *Discret. Math.* **109**(1–3), 117–126 (1992)
19. Hodges, W., et al.: *A shorter model theory*. Cambridge University Press (1997)
20. Ladner, R.E.: On the structure of polynomial time reducibility. *J. ACM (JACM)* **22**(1), 155–171 (1975)
21. Laskowski, M.C., et al.: Mutually algebraic structures and expansions by predicates. *J. Symb. Log.* **78**(1), 185–194 (2013)
22. Mottet, A., Bodirsky, M.: A dichotomy for first-order reducts of unary structures. *Logical Methods in Computer Science* **14** (2018)
23. Zhuk, D.: A proof of csp dichotomy conjecture. In: *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 331–342. IEEE (2017)
24. Zhuk, D.: A proof of the CSP dichotomy conjecture. *J. ACM (JACM)* **67**(5), 1–78 (2020)



# Circle Graph Isomorphism in Almost Linear Time

Vít Kalisz<sup>1</sup>, Pavel Klavík<sup>1</sup>, and Peter Zeman<sup>2(✉)</sup>

<sup>1</sup> OrgPad, Prague, Czech Republic  
{kalisz,klavik}@orgpad.com

<sup>2</sup> Institut de Mathématiques, Université de Neuchâtel, Neuchâtel, Switzerland  
zeman.peter.sk@gmail.com  
<https://orgpad.com>

**Abstract.** Circle graphs are intersection graphs of chords of a circle. In this paper, we present a new algorithm for the circle graph isomorphism problem running in time  $\mathcal{O}((n+m)\alpha(n+m))$  where  $n$  is the number of vertices,  $m$  is the number of edges and  $\alpha$  is the inverse Ackermann function. Our algorithm is based on the minimal split decomposition [Cunnigham, 1982] and uses the state-of-art circle graph recognition algorithm [Gioan, Paul, Tedder, Corneil, 2014] in the same running time. It improves the running time  $\mathcal{O}(nm)$  of the previous algorithm [Hsu, 1995] based on a similar approach.

**Keywords:** Circle graphs · Graph isomorphism problem · Graph canonization · Split decomposition

## 1 Introduction

For graphs  $G$  and  $H$ , a bijection  $\pi : V(G) \rightarrow V(H)$  is called an *isomorphism* if  $uv \in E(G) \iff \pi(u)\pi(v) \in E(H)$ . Testing isomorphism of graphs in polynomial time is a major open problem in theoretical computer science.

For a graph  $G$ , a *circle representation*  $\mathcal{R}$  of a graph  $G$  is a collection of sets  $\{\langle v \rangle : v \in V(G)\}$  such that each  $\langle v \rangle$  is a chord of some fixed circle, and  $\langle u \rangle \cap \langle v \rangle \neq \emptyset$  if and only if  $uv \in E(G)$ . Observe that  $\mathcal{R}$  is determined by the circular word giving the clockwise order of endpoints of the chords in which  $uv \in E(G)$  if and only if their endpoints alternate as  $uvuv$  in this word. A graph is called a *circle graph* if and only if it has a circle representation; see Fig. 1.

Circle graphs, introduced by Even and Itai [9], are related to Gauss words [11], matroid representations [4, 10], and rank-width [18]. The complexity of recognition of circle graphs was a long-standing open problem, resolved in mid-1980s [3, 12, 17]. Currently, the fastest recognition algorithm [13] runs in almost linear time. In this paper, we use this recognition algorithm as a subroutine and solve the graph isomorphism problem of circle graphs in the same running time.

---

P. Zeman—Was supported by the Swiss National Science Foundation project PP00P2-202667. While at Department of Applied Mathematics, Faculty of Mathematics and Physics, Charles University, Peter Zeman was supported by GAČR 20-15576S.

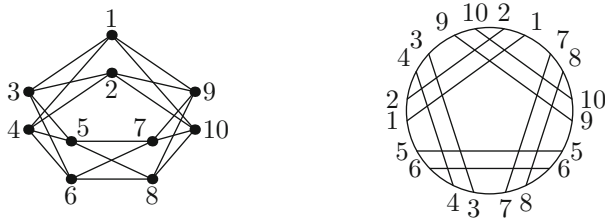
**Theorem 1.** *The graph isomorphism problem of circle graphs and the canonization problem of circle graphs can be solved in time  $\mathcal{O}((n + m) \cdot \alpha(n + m))$ , where  $n$  is the number of vertices,  $m$  is the number of edges, and  $\alpha$  is the inverse Ackermann function. Further, if circle representations are given as a part of the input, the running time improves to  $\mathcal{O}(n + m)$ .*

Two circle representation are isomorphic if by relabeling the endpoints we get identical circular orderings. In Sect. 4, we show that isomorphism of circle representations can be tested in time  $\mathcal{O}(n)$ . When circle graphs  $G$  and  $H$  have isomorphic circle representations  $\mathcal{R}_G$  and  $\mathcal{R}_H$ , clearly  $G \cong H$ . But in general, the converse does not hold since a circle graph may have many non-isomorphic circle representations.

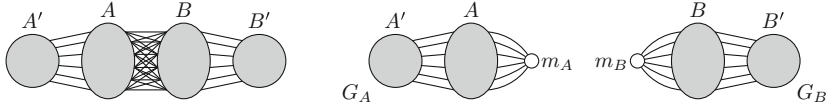
The main tool is the split decomposition which is a recursive process decomposing a graph into several indecomposable graphs called *prime graphs*. Each split decomposition can be described by a split tree whose nodes are the prime graphs on which the decomposition terminates. The key property is that the initial graph is a circle graph if and only if all prime graphs are circle graphs. Further, each prime circle graph has a unique representation up to reversal, so isomorphism for them can be tested in  $\mathcal{O}(n)$ , using the approach described in Sect. 4.

It might be tempting to reduce the isomorphism problem of circle graphs to the isomorphism problem of split trees. Unfortunately, a graph may posses many different split decompositions corresponding to non-isomorphic split trees. The seminal paper of Cunningham [5, Theorem 3] shows that for every connected graph, there exists a minimal split decomposition; this result was also proven in [6, Theorem 11]. The split tree associated to the minimal split decomposition is then also unique and it follows that the isomorphism problem of circle graphs reduces to the isomorphism problem of minimal split trees.

This approach was used by Hsu [15] to solve the graph isomorphism problem of circle graphs in time  $\mathcal{O}(nm)$ . He actually concentrates on circular-arc graphs, which are intersection graphs of circular arcs, and builds a decomposition technique which generalizes minimal split decomposition. The main results are recognition and graph isomorphism algorithms for circular-arc graphs running in  $\mathcal{O}(nm)$ . Unfortunately, a mistake in this general decomposition technique



**Fig. 1.** A circle graph and one of its circle representations corresponding to the circular word 10, 2, 1, 7, 8, 10, 9, 5, 6, 8, 7, 3, 4, 6, 5, 1, 2, 4, 3, 9.



**Fig. 2.** On the left, a split in  $G$  between  $A$  and  $B$ . On the right, application of this split produces graphs  $G_A$  and  $G_B$  with newly created marker vertices denoted by big white circles.

was pointed out by [7]. This mistake does not affect the graph isomorphism algorithm for circle graphs. Moreover, as pointed out in [7, page 180], the complexity of Hsu’s algorithm improves to  $\mathcal{O}((n + m) \cdot \alpha(n + m))$  if the fastest recognition algorithm of circle graphs [13] is employed. In [7, page 180], it is stated: “If chord models are given as an input, then the running time of the isomorphism test can be reduced to  $\mathcal{O}(n + m)$  using techniques similar to those used in [16] and in our paper”. This paper addresses this remark.

A well-known subclass of circle graphs are proper circular-arc graphs, which are intersection graphs of circular arcs such that no arc is properly contained in another one. The methods used in the linear-time algorithm testing isomorphism of proper circular-arc graphs, given in [16], are similar to the case of prime circle graphs (see Sect. 4). In particular, co-bipartite circular-arc graphs have unique representations and, in this case, the problem can be reduced to finding minimal circular string.

## 2 Minimal Split Decomposition and Split Trees

In this section, we describe several known properties of split decompositions and split trees. We assume that all graphs are connected, otherwise split decomposition is applied independently on each component.

**Splits.** For a graph  $G$ , a *split* is a partition  $(A, B, A', B')$  of  $V(G)$  such that:

- For every  $a \in A$  and  $b \in B$ , we have  $ab \in E(G)$ .
- There are no edges between  $A'$  and  $B \cup B'$ , and between  $B'$  and  $A \cup A'$ .
- Both sides have at least two vertices:  $|A \cup A'| \geq 2$  and  $|B \cup B'| \geq 2$ .

See Fig. 2 on the left. In other words, the cut between  $A$  and  $B$  is the complete bipartite graph. A split in a graph can be found in polynomial time [20]. Graphs containing no splits are called *prime graphs*. Since the sets  $A$  and  $B$  already uniquely determine the split, we call it the *split between  $A$  and  $B$* .

We can *apply* a split between  $A$  and  $B$  to divide the graph  $G$  into two graphs  $G_A$  and  $G_B$  defined as follows. The graph  $G_A$  is created from  $G[A \cup A']$  together with a new *marker vertex*  $m_A$  adjacent exactly to the vertices in  $A$ . The graph  $G_B$  is defined analogously for  $B$ ,  $B'$  and  $m_B$ . See Fig. 2 on the right.

**Split Decomposition and Split Trees.** A split decomposition  $D$  of  $G$  is a sequence of splits defined as follows. At the beginning, we start with the graph  $G$ . In the  $k$ -th step, we have graphs  $G_1, \dots, G_k$  and we apply a split on some  $G_i$ , dividing it into two graphs  $G'_i$  and  $G''_i$ . The next step then applies to one of the graphs  $G_1, \dots, G_{i-1}, G'_i, G''_i, G_{i+1}, \dots, G_k$ , and so on.

A split decomposition can be captured by a graph-labeled tree  $T$ . The vertices of  $T$  are called nodes to distinguish them from the vertices of  $G$  and from the added marker vertices, and nodes correspond to subsets of these vertices. To simplify the definition of graph isomorphism of graph-labeled trees, we give a slightly different formal definition in which  $T$  is not necessarily a tree.

**Definition 1.** A graph-labeled tree  $T$  is a graph  $(V, E)$  with  $E = E_N \dot{\cup} E_T$  where  $E_N$  are called the normal edges and  $E_T$  are called the tree edges. A node is a connected component of  $(V, E_N)$ . There are no tree edges between the vertices of one node and no vertex is incident to two tree edges. The incidence graph of nodes must form a tree. The size of  $T$  is  $|V| + |E|$ .

A graph-labeled tree  $T$  might not be a tree, but the underlying structure of tree edges forms a tree of nodes. The vertices of  $T$  incident to tree edges are called *marker vertices*.

A split decomposition  $D$  of  $G$  is represented by the following graph-labeled tree  $T$  called the *split tree*  $T$  of  $D$  (or a split tree  $T$  of  $G$ ). Initially,  $T$  consists of a single node equal to  $G$ . At each step,  $D$  applies a split on one node  $N$  of  $T$ . This node is replaced by two new nodes  $N_A$  and  $N_B$  while the tree edges incident to  $N$  are preserved in  $N_A$  and  $N_B$  and the marker vertices  $m_A$  and  $m_B$  are further adjacent by a newly formed tree edge. Figure 3 shows an example. It can be observed that the total size of every split tree is  $\mathcal{O}(n + m)$  where  $n$  is the number of vertices and  $m$  is the number of edges of the original graph.

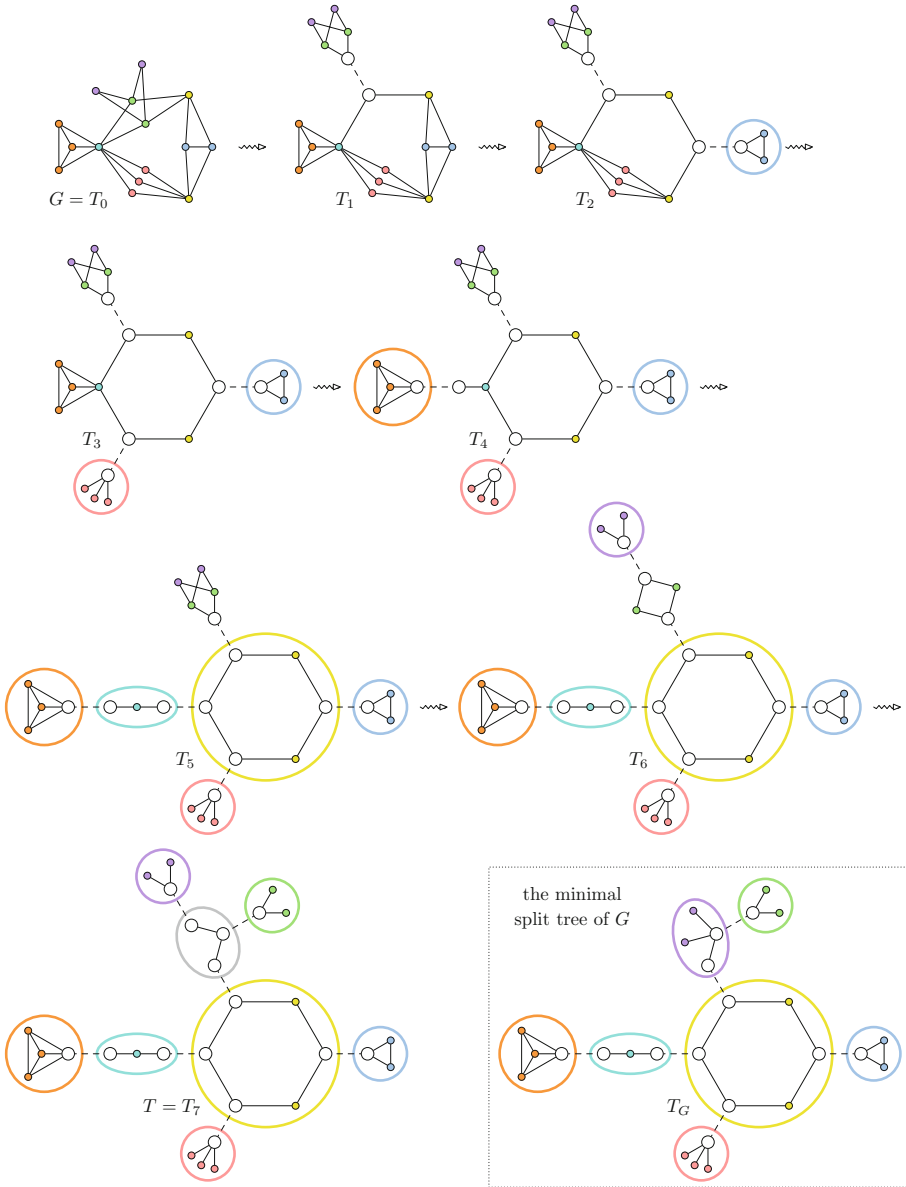
From a split tree  $T$ , the original graph  $G$  can be reconstructed by *joining neighboring nodes*. For a tree edge  $m_A m_B$ , we remove  $m_A$  and  $m_B$  while adding all edges  $uv$  for  $u \in N(m_A)$  and  $v \in N(m_B)$ .

**Recognition of Circle Graphs.** A split decomposition can be applied to recognize circle graphs. The key is the following observation.

**Lemma 1.** A graph is a circle graph if and only if both  $G_A$  and  $G_B$  are circle graphs.

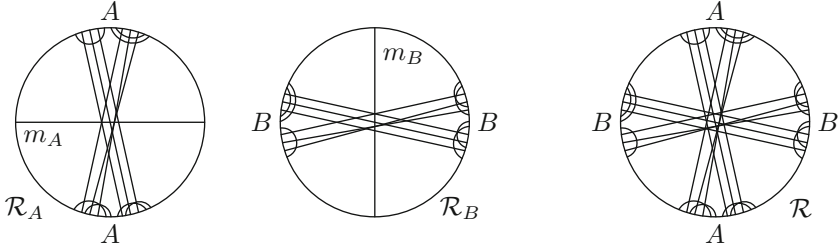
The proof is illustrated in Fig. 4, which can be easily formalized; see for example [21]. A prime circle graph has a unique circle representation up to reversal [8] which can be constructed in polynomial time [13].

**Minimal Split Decomposition.** A graph is called *degenerate* if it is the complete graph  $K_n$  or the star  $S_n$ . Suppose that we have a split decomposition  $D$  ending on prime graphs. Its split tree is not uniquely determined, for instance degenerate graphs have many different split trees. Cunnigham [5] resolved this issue by terminating the split decomposition not only on prime graphs, but also on the degenerate graphs.



**Fig. 3.** An example of a split tree  $T$  of a split decomposition  $D$  terminating with highlighted prime and degenerate graphs (see the definition below). The split decomposition  $D$  is not minimal: the gray and purple stars can be joined in  $T$  to form the minimal split tree  $T_G$  in the box. (Color figure online)





**Fig. 4.** On the left, circle representations  $\mathcal{R}_A$  and  $\mathcal{R}_B$  of graphs  $G_A$  and  $G_B$ . They are combined into a circle representation  $\mathcal{R}$  of  $G$ .

Cunningham [5] introduced the notion of a *minimal split decomposition*. A split decomposition is minimal if the corresponding split tree has all nodes as prime and degenerate graphs, and joining any two neighboring nodes creates a non-degenerate graph.

**Theorem 2 (Cunningham [5], Theorem 3).** *For a connected graph  $G$ , the split tree of a minimal split decomposition terminating on prime and degenerate graphs is uniquely determined.*

The split tree of a minimal split decomposition of  $G$  is called the *minimal split tree* of  $G$  and it is denoted  $T_G$ .

It was stated in [14, Theorem 2.17] that a split decomposition is minimal if the corresponding split tree has no two neighboring nodes such that

- either both are complete,
- or both are stars and the tree edge joining these stars is incident to exactly one of their central vertices.

The reason is that such neighboring nodes can be joined into a complete node or a star node, respectively. Therefore, the minimal split tree can be constructed from an arbitrary split tree by joining neighboring complete graphs and stars. For instance, the split tree  $T$  in Fig. 3 is not minimal since the purple and gray stars can be joined, creating the minimal split tree  $T_G$ .

Cunningham’s definition of a minimal split decomposition is with respect to inclusion. Since the minimal split decomposition is uniquely determined, it is equivalently the split decomposition terminating on prime and degenerate graphs using the least number of splits.

**Computation of Minimal Split Trees.** The minimal split tree can be computed in time  $\mathcal{O}(n+m)$  using the algorithm of [8]. For the purpose of this paper, we use the following slower algorithm since it also computes the unique circle representations of encountered prime circle graphs:

**Theorem 3 (Gioan et al. [13,14]).** *The minimal split tree  $T_G$  of a circle graph  $G$  can be computed in time  $\mathcal{O}((n+m) \cdot \alpha(n+m))$  where  $n$  is the number*

of vertices,  $m$  is the number of edges, and  $\alpha$  is the inverse Ackermann function. Further, the algorithm also computes the unique circle representation of each prime circle node of  $T_G$ .

**Graph isomorphism via Minimal Split Decompositions.** Let  $T$  and  $T'$  be two graph-labeled trees. An *isomorphism*  $\pi : T \rightarrow T'$  is an isomorphism which maps normal edges to normal edges and tree edges to tree edges. Notice that  $\pi$  maps nodes of  $T$  to isomorphic nodes of  $T'$  while preserving tree edges.

We use minimal split trees to test graph isomorphism of circle graphs (for a proof see the Appendix):

**Lemma 2.** *Two connected graphs  $G$  and  $H$  are isomorphic if and only if the minimal split trees  $T_G$  and  $T_H$  are isomorphic.*

*Proof.* Let  $T'_G$  and  $T'_H$  be any split trees of  $G$  and  $H$ , respectively, and  $\pi : T'_G \rightarrow T'_H$  be an isomorphism. We want to show that  $G \cong H$ . Choose an arbitrary tree edge  $e = m_A m_B$  in  $T'_G$ , we know that  $\pi(e) = \pi(m_A)\pi(m_B)$  is a tree edge in  $T'_H$ . We join  $T'_G$  over  $e$  and  $T'_H$  over  $\pi(e)$ . We get that the restriction  $\pi|_{T'_G \setminus \{m_A, m_B\}}$  is an isomorphism of the constructed graph-labeled trees. By repeating this process, we get single nodes isomorphic graph-labeled trees which are  $G$  and  $H$  respectively. So  $G \cong H$ .

For the other implication, suppose that  $\pi : G \rightarrow H$  is an isomorphism. Let  $D_G$  be a minimal split decomposition, constructing the minimal split tree  $T_G$ . We use  $\pi$  to construct a split decomposition  $D_H$  and a split tree  $T_H$  of  $H$  such that  $T_G \cong T_H$ . Before any splits, the trees  $T_G^0 \cong G$  and  $T_H^0 \cong H$  are isomorphic. Suppose that  $T_G^k \cong T_H^k$  and  $D_G$  then uses a split between  $A$  and  $B$  in some node  $N$ . Then  $D_H$  will use the split between  $\pi(A)$  and  $\pi(B)$ , and since  $\pi$  is an isomorphism, it is a valid split in  $\pi(N)$ . We construct  $T_G^{k+1}$  by splitting  $N$  into two nodes  $N_A$  and  $N_B$  and adding marker vertices  $m_A$  and  $m_B$ , and similarly for  $T_H^{k+1}$  with marker vertices  $m_{\pi(A)}$  and  $m_{\pi(B)}$ . We extend  $\pi$  to an isomorphism from  $T_G^{k+1}$  to  $T_H^{k+1}$  by setting  $\pi(m_A) = m_{\pi(A)}$  and  $\pi(m_B) = m_{\pi(B)}$ . Therefore, the resulting split trees  $T_G$  and  $T_H$  are isomorphic. By Theorem 2, the minimal split tree of  $H$  is uniquely determined, so it has to be isomorphic to the constructed  $T_H$ .  $\square$

### 3 Canonization of Graph-Labeled Trees

In the rest of the paper, we work with colored graphs and isomorphisms are required to be color-preserving. Colors are represented as non-negative integers.

**Definition of Canonization.** Let  $G$  be a colored graph with  $n$  vertices with colors in the range  $0, \dots, n-1$  and  $m$  edges. An *encoding*  $\varepsilon(G)$  of  $G$  is a sequence of non-negative integers. The encoding  $\varepsilon(G)$  is *linear* if it contains at most  $\mathcal{O}(n+m)$  integers, each in range  $0, \dots, n-1$ . We denote the class of all encodings by  $\mathcal{E}$ . For a class of graphs  $\mathcal{C}$ , a *linear canonization* is some function  $\gamma : \mathcal{C} \rightarrow \mathcal{E}$  such that  $\gamma(G)$  is a linear encoding of  $G$  and for  $G, H \in \mathcal{C}$ , we have  $G \cong H$  if and only if  $\gamma(G) = \gamma(H)$ .

**Fast Lexicographic Sorting.** Since we want to sort these encodings lexicographically, we frequently use the following well-known algorithm:

**Lemma 3 (Aho, Hopcroft, and Ullman [1], Algorithm 3.2, p. 80).** *It is possible to lexicographically sort sequences of numbers  $0, \dots, t-1$  of arbitrary lengths in time  $\mathcal{O}(\ell + t)$  where  $\ell$  is the total length of these sequences.*

It is easy to modify the above algorithm to get the same running time when all numbers belong to  $\{0, 1, 2, \dots, t+1, s, s+1, \dots, s+t-1\}$ .

**Canonization Algorithm.** In the rest of this section, we are going to describe the following meta-algorithm.

**Lemma 4.** *Let  $\mathcal{C}$  be a class of graphs and  $\mathcal{T}$  be a class of graph-labeled trees whose nodes belong to  $\mathcal{C}$ . Suppose that we can compute a linear-space canonization  $\gamma$  of colored graphs in  $\mathcal{C}$  in time  $f(n+m)$  where  $n$  is the number of vertices,  $m$  is the number of edges, and  $f$  is convex. Then we can compute a linear-space canonization  $\tilde{\gamma}$  of graph-labeled trees from  $\mathcal{T}$  in time  $\mathcal{O}(n+m+f(n+m))$ .*

Let  $T \in \mathcal{T}$  be a graph-labeled tree. Recall that every tree has either a central vertex or a central edge. We may assume that  $T$  is rooted at the central node: If a tree edge is central, we insert another node having a single vertex. Also, we orient all tree edges towards the root. For a node  $N$ , we denote by  $T[N]$  the graph-labeled subtree induced by  $N$  and all descendants of  $N$ . Initially, we color all marker vertices by the color 0 and all other vertices by the color 1. Throughout the algorithm, only marker vertices change colors.

The  $k$ -th layer in  $T$  is formed by all nodes of the distance  $k$  from the root. Notice that every isomorphism from  $T$  to  $T'$  maps, for every  $k$ , the  $k$ -th layer of  $T$  to the  $k$ -th layer of  $T'$ . Also, every node  $N$  aside the root is incident to exactly one out-going tree edge whose incident marker vertex outside  $N$  is called the *parent marker vertex* of  $N$ .

The algorithm starts from the bottom layer of  $T$  and process the layers towards the root. When a node  $N$  is processed, we assign a color  $c$  to  $N$ . This color  $c$  corresponds to a certain linear encoding  $\gamma'(N)$  which is created by modifying  $\gamma(N)$ . Further, we store the mapping  $\varepsilon$  from colors to these linear encodings, so  $\varepsilon(c) = \gamma'(N)$ . The assigned colors have the property that two nodes  $N$  and  $N'$  have the same assigned color if and only if the rooted graph-labeled subtrees induced by  $N$  and all the nodes below and by  $N'$  and all the nodes below, respectively, are isomorphic. We remove the node  $N$  from  $T$  and assign its color to the parent marker vertex of  $N$ . Also notice that when a non-root node is processed, all marker vertices except for one have colors different from 0, and for the root node, all marker vertices have colors different from 0.

Let  $N_1, \dots, N_k$  be the nodes of the currently processed layer such that each vertex in these nodes has one color from  $\{0, 1, \dots, t+1, s, s+1, \dots, s+t-1\}$ . For each node  $N_i$ , we want to use the canonization subroutine to compute the linear encoding  $\gamma(N_i)$ . But the assumptions require that for  $\ell$  vertices in  $N_i$ ,

all colors are in range  $0, \dots, \ell - 1$ , but we might have  $s \gg \ell$ . We can avoid this by renumbering the colors since at most  $\ell$  different colors are used on  $N_i$ . Suppose that exactly  $c_i$  different colors are used in  $N_i$ , and we define the injective mapping  $\varphi_{N_i} : \{0, 1, \dots, c_i - 1\} \rightarrow \{0, 1, \dots, t + 1, s, s + 1, \dots, s + t - 1\}$  such that the smallest used color is  $\varphi_{N_i}(0)$ , the second smallest is  $\varphi_{N_i}(1)$ , and so on till  $\varphi_{N_i}(c_i - 1)$ . The renumbering of colors on  $N_i$  is given by the inverse  $\varphi_{N_i}^{-1}$ .

After renumbering, the algorithm runs the canonization subroutine to compute the linear encodings  $\gamma(N_1), \dots, \gamma(N_k)$ . We create the modified linear encoding  $\gamma'(N_i)$  by pre-pending  $\gamma(N_i)$  with the sequence  $c_i, \varphi_{N_i}(0), \varphi_{N_i}(1), \dots, \varphi_{N_i}(c_i - 1)$  where  $c_i$  is the number of different colors used in  $N_i$ . The algorithm lexicographically sorts these modified encodings  $\gamma'(N_1), \dots, \gamma'(N_k)$  using Lemma 3. Next, we assign the color  $s + t$  to the nodes having the smallest encodings, the color  $s + t + 1$  to the nodes having the second smallest encodings, and so on. For every node  $N_i$ , we remove it and set the color of the parent marker vertex of  $N_i$  to the color assigned to  $N_i$ .

Suppose that the root node  $N$  has the color  $c$  assigned, so throughout the algorithm, we have used the colors  $0, \dots, c$ . The computed linear encoding of  $T$  is an encoded concatenation of  $\varepsilon(2), \varepsilon(3), \dots, \varepsilon(c)$ .

Lemma 4 follows from the next two lemmas.

**Lemma 5.** *The described algorithm produces a correct linear canonization  $\tilde{\gamma}$  of graph-labeled trees in  $\mathcal{T}$ , i.e., for  $T, T' \in \mathcal{T}$ , we have  $T \cong T'$  if and only if  $\tilde{\gamma}(T) = \tilde{\gamma}(T')$ .*

*Proof.* Let  $\pi : T \rightarrow T'$  be an isomorphism. We prove by induction from the bottom layer to the root that  $\gamma'(N) = \gamma'(\pi(N))$ . Suppose that we are processing the  $\ell$ -th layer of  $T$  and  $T'$  and all previously used colors have the same assigned encodings in  $T$  and  $T'$ . For every node  $N$  of the  $\ell$ -th layer of  $T$ , then  $\pi(N)$  belongs to the  $\ell$ -th layer of  $T'$ . We argue that  $\pi$  also preserves the colors of  $N$ . Since  $\pi$  is an isomorphism of graph-labeled trees, it maps marker vertices to marker vertices and non-marker vertices to non-marker vertices. Let  $N_1, \dots, N_k$  be the children of  $N$ . By the induction hypothesis, we have  $\gamma'(N_i) = \gamma'(\pi(N_i))$ , so the same colors are assigned to  $N_i$  and  $\pi(N_i)$ . Therefore,  $\pi$  preserves the colors of  $N$ , and this property still holds after renumbering the colors by  $\varphi_N^{-1} = \varphi_{\pi(N)}^{-1}$ . Therefore, we have  $\gamma(N) = \gamma(\pi(N))$  and thus  $\gamma'(N) = \gamma'(\pi(N))$ . Thus, the lexicographic sorting of the nodes in the  $\ell$ -th layer of  $T$  and of  $T'$  is the same, so  $N$  and  $\pi(N)$  have the same color assigned. Finally, the encodings  $\varepsilon(2), \dots, \varepsilon(c)$  are the same in  $T$  and  $T'$ , so  $\tilde{\gamma}(T) = \tilde{\gamma}(T')$ .

For the other implication, we show that a graph-labeled tree  $T'$  isomorphic to  $T$  can be reconstructed from  $\tilde{\gamma}(T)$ . We construct the root node  $N$  from  $\varepsilon(c)$ . Since  $\gamma$  is a canonization of  $\mathcal{C}$ , we obtain  $N$  by applying  $\gamma^{-1}$ . Next, we invert the recoloring by applying  $\varphi(N)$  on the colors of  $N$ . Next, we consider each marker vertex in  $N$ . If it has some color  $c_i$ , we use  $\varepsilon(c_i)$  to construct a child node  $N'$  of  $N$  exactly as before. We proceed in this way till all nodes are expanded and only the colors 0 and 1 remain. It is easy to prove by induction that  $T \cong T'$  since each  $\varepsilon(c_i)$  uniquely determines the corresponding subtree in  $T$ .  $\square$

**Lemma 6.** *The described algorithm runs in time  $\mathcal{O}(n + m + f(n + m))$ .*

*Proof.* When we run the canonization subroutine on a node having  $n'$  vertices and  $m'$  edges, it has all colors in range  $0, \dots, n' - 1$ , so we can compute its linear encoding in time  $f(n' + m')$ . Since  $f$  is convex and the canonization subroutine runs on each node exactly once, the total time spend by this subroutine is bounded by  $f(n + m)$ .

The total count of used colors is clearly bounded by  $n$  and each layer uses different colors except for 0 and 1. Consider a layer with nodes  $N_1, \dots, N_k$  having  $\ell$  vertices and  $\ell'$  edges in total. All these nodes use at most  $\ell$  different colors. Therefore, the modified encodings  $\gamma'(N_1), \dots, \gamma'(N_k)$  consisting of integers from  $\{0, 1, 2, \dots, \ell + 1, s, s + 1, \dots, s + \ell - 1\}$ , for some value  $s$ , and are of the total length  $\mathcal{O}(\ell + \ell')$ . Therefore, lexicographic sorting of these modified encodings can be done in time  $\mathcal{O}(\ell + \ell')$ , and this sorting takes total time  $\mathcal{O}(n + m)$  for all layers of  $T$ . The total running time of the algorithm is  $\mathcal{O}(n + m + f(n + m))$ .  $\square$

## 4 Canonization of Prime and Degenerate Circle Graphs

Let  $\mathcal{C}$  be the class of colored prime and degenerate circle graphs. Recall that all nodes of minimal split trees of connected circle graphs belong to  $\mathcal{C}$ . To apply the meta-algorithm of Lemma 4, we need to show that the linear canonization  $\gamma$  of  $\mathcal{C}$  can be computed in time  $\mathcal{O}(n + m)$  where  $n$  is the number of vertices and  $m$  is the number of edges.

**Linear Canonizations of Degenerate Graphs.** For a colored complete graph  $G = K_n$ , we sort its colors using bucket sort in time  $\mathcal{O}(n)$ , so the vertices have the colors  $c_1 \leq c_2 \leq \dots \leq c_n$ . The computed linear canonization  $\gamma(G)$  is  $0, c_1, c_2, \dots, c_n$ .

For a star  $G = S_n$ , we sort the colors of leaves using bucket sort in time  $\mathcal{O}(n)$ , so they have the colors  $c_1 \leq c_2 \leq \dots \leq c_n$ , while the center has the color  $c_0$ . The computed linear canonization  $\gamma(G)$  is  $1, c_0, c_1, c_2, \dots, c_n$ .

**Linear Encodings of Colored Cycles.** As a subroutine, we need to find a canonical form of a colored cycle. To do this, it suffices to find the lexicographically minimal rotation of a circular string. This can be done using  $\mathcal{O}(n)$  comparisons over some alphabet  $\Sigma$  [2, 19].

**Linear Canonizations of Circle Representations.** Let  $G$  be an arbitrary colored circle graph on  $n$  vertices together with an arbitrary circle representation  $\mathcal{R}$ . The standard way to describe  $\mathcal{R}$  is to arbitrarily order the vertices  $1, \dots, n$  and to give a circular word  $\omega$  consisting of  $2n$  integers from  $1, \dots, n$ , each appearing exactly twice, in such a way that the occurrences of  $i$  and  $j$  alternate (i.e., appear as  $ijij$ ) if and only if  $ij \in E(G)$ . This circular word describes the ordering of the endpoints of the chords in, say, the clockwise direction.

Let  $G$  and  $H$  be two colored circle graphs on  $n$  vertices labeled  $1, \dots, n$  with circle representations  $\mathcal{R}_G$  and  $\mathcal{R}_H$  represented by  $\omega_G = \omega_G^1, \dots, \omega_G^{2n}$  and  $\omega_H = \omega_H^1, \dots, \omega_H^{2n}$ . We say that  $\mathcal{R}_G \cong \mathcal{R}_H$  if and only if there exists a bijection  $\pi : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$  such that

- the vertices  $i$  in  $G$  and  $\pi(i)$  in  $H$  have identical colors, and
- the circular words  $\omega_H$  and  $\pi(\omega_G^1), \dots, \pi(\omega_G^{2n})$  are identical.

Notice that when  $\mathcal{R}_G \cong \mathcal{R}_H$ , necessarily  $G \cong H$ , but in general the converse is not true. We want to construct a linear canonization  $\gamma$  such that  $\mathcal{R}_G \cong \mathcal{R}_H$  if and only if  $\gamma(\mathcal{R}_G) = \gamma(\mathcal{R}_H)$ .

To this end, we consider a different encoding of the representation which is invariant on rotation. For each of  $2n$  endpoints  $e_1, \dots, e_{2n}$ , we store two numbers:

- The color  $c_i \in \{0, \dots, n - 1\}$  of the vertex of the chord corresponding to  $e_i$ .
- The number of endpoints  $g_i$  in the clockwise direction between  $e_i$  and the other endpoint corresponding to the same chord. We have  $g_i \in \{0, \dots, 2n - 2\}$  and when the  $e_i$  and  $e_j$  correspond one chord, then  $g_i + g_j = 2n - 2$ .

To distinguish  $c_i$  from  $g_i$ , we increase all  $c_i$  by  $2n - 1$ , so  $c_i \in \{2n - 1, \dots, 3n - 2\}$ . Then we may consider the circular word  $\lambda_G = g_1, c_1, g_2, c_2, \dots, g_{2n}, c_{2n}$  of length  $4n$ .

**Lemma 7.** *Let  $G$  and  $H$  be two colored circle graphs with representations  $\mathcal{R}_G$  and  $\mathcal{R}_H$ . We have  $\mathcal{R}_G \cong \mathcal{R}_H$  if and only if the circular words  $\lambda_G$  and  $\lambda_H$  are identical.*

*Proof.* If  $\mathcal{R}_G \cong \mathcal{R}_H$ , there exists an index  $k \in \{0, \dots, 2n - 1\}$  such that rotating the representation  $\mathcal{R}_G$  by  $k$  endpoints produces  $\mathcal{R}_H$ . When  $\lambda_G = g_1, c_1, \dots, g_{2n}, c_{2n}$  and  $\lambda_H = g'_1, c'_1, \dots, g'_{2n}, c'_{2n}$ , it cyclically holds that  $g_i = g'_{i+k}$  and  $c_i = c'_{i+k}$ . So the circular words  $\lambda_G$  and  $\lambda_H$  are identical.

For the other implication, observe that the circle representation  $\mathcal{R}_G$  and the circle graph  $G$  can be reconstructed from  $\lambda_G$ . If  $\lambda_G$  and  $\lambda_H$  are identical, we reconstruct isomorphic representations  $\mathcal{R}_G$  and  $\mathcal{R}_H$ . □

**Lemma 8.** *We can compute the linear encoding  $\gamma$  of colored circle representations in time  $\mathcal{O}(n)$ .*

*Proof.* For a representation  $\mathcal{R}_G$ , we can clearly compute  $\lambda_G$  in time  $\mathcal{O}(n)$ . Next, we apply to  $\lambda_G$  a cycle canonization algorithm [2, 19] which computes  $\gamma(\mathcal{R}_G)$  in time  $\mathcal{O}(n)$ . □

**Linear Canonization of Prime Circle Graphs.** Let  $G$  be a prime circle graph. It has at most two different representations  $\mathcal{R}_G$  and  $\mathcal{R}'_G$  where one is the reversal of the other. Using Lemma 8, we compute their linear encodings  $\lambda_G$  and  $\lambda'_G$ . As the linear encoding  $\gamma(G)$ , we chose the lexicographically smallest of  $\lambda_G$  and  $\lambda'_G$ , prepended with the value 2. Clearly, colored prime circle graphs  $G$  and  $H$  are isomorphic if and only if  $\gamma(G) = \gamma(H)$ .

By putting the results of this section together, we get the following:

**Lemma 9.** *We can compute linear canonization of colored prime circle graphs and degenerate graphs in time  $\mathcal{O}(n)$ .*

## 5 Proof of Theorem 1

In this section, we combine the presented results to show that a linear canonization  $\gamma$  of circle graphs can be computed in time  $\mathcal{O}((n+m) \cdot \alpha(n+m))$ . This algorithm clearly implies Theorem 1 since circle graphs  $G$  and  $H$  are isomorphic if and only if  $\gamma(G) = \gamma(H)$ .

Suppose that  $G$  is a connected circle graph. We apply the algorithm of Theorem 3 to compute the minimal split decomposition  $T_G$  of  $G$  and the unique circle representation for each prime circle graph (up to reversal). We halt if some circle representations does not exist since  $G$  is not a circle graph. The total running time of preprocessing is  $\mathcal{O}((n+m) \cdot \alpha(n+m))$ , and the remainder of the algorithm runs in time  $\mathcal{O}(n+m)$ , so this step is the bottleneck. Next, we use Lemmas 9 and 4 to compute a linear canonization  $\gamma(T_G)$  and we put  $\gamma(G) = \gamma(T_G)$ .

Suppose that the circle graph  $G$  is disconnected, and let  $G_1, \dots, G_k$  be its connected components. We compute their linear encodings  $\gamma(G_1), \dots, \gamma(G_k)$ , lexicographically sort them in time  $\mathcal{O}(n+m)$  using Lemma 3, and output them in  $\gamma(G)$  sorted as a sequence. The total running time is  $\mathcal{O}((n+m) \cdot \alpha(n+m))$ .

When the input also gives a circle representation  $\mathcal{R}$ , we can avoid using Theorem 3. Instead, we compute a split decomposition and the corresponding split tree in time  $\mathcal{O}(n+m)$  using [8]. We can easily modify this split tree into the minimal split tree by joining neighboring complete vertices and stars as discussed in Sect. 2. For each prime node  $N$ , we obtain its unique circle representation by restricting  $\mathcal{R}$  to the vertices of  $N$ . Since the avoided algorithm of Theorem 3 was the bottleneck, we get the total running time  $\mathcal{O}(n+m)$ .

## References

1. Aho, A.V., Hopcroft, J.E., Ullman, J.D.: The design and analysis of computer algorithms. Addison-Wesley Publishing Company (1974)
2. Booth, K.S.: Lexicographically least circular substrings. *Inf. Process. Lett.* **10**(4–5), 240–242 (1980)
3. Bouchet, A.: Reducing prime graphs and recognizing circle graphs. *Combinatorica* **7**(3), 243–254 (1987)
4. Bouchet, A.: Unimodularity and circle graphs. *Discret. Math.* **66**(1–2), 203–208 (1987)
5. Cunningham, W.H.: Decomposition of directed graphs. *SIAM J. Algebraic Discrete Methods* **3**(2), 214–228 (1982)
6. Cunningham, W.H., Edmonds, J.: A combinatorial decomposition theory. *Can. J. Math.* **32**(3), 734–765 (1980)
7. Curtis, A.R., et al.: Isomorphism of graph classes related to the circular-ones property. *Discrete Math. Theor. Comput. Sci.* **15**(1), 157–182 (2013)

8. Dahlhaus, E.: Parallel algorithms for hierarchical clustering and applications to split decomposition and parity graph recognition. *J. Algorithms* **36**(2), 205–240 (1998)
9. Even, S., Itai, A.: Queues, stacks, and graphs. In: Kohavi, Z., Paz, A. (eds.) *Theory of Machines and Computations*, pp. 71–76 (1971)
10. de Fraysseix, H.: Local complementation and interlacement graphs. *Discret. Math.* **33**(1), 29–35 (1981)
11. de Fraysseix, H., de Mendez, P.O.: On a characterization of gauss codes. *Discrete Comput. Geom.* **22**(2), 287–295 (1999)
12. Gabor, C.P., Supowit, K.J., Hsu, W.: Recognizing circle graphs in polynomial time. *J. ACM* **36**(3), 435–473 (1989)
13. Gioan, E., Paul, C., Tedder, M., Corneil, D.: Practical and efficient circle graph recognition. *Algorithmica* **69**(4), 759–788 (2014). <https://doi.org/10.1007/s00453-013-9745-8>
14. Gioan, E., Paul, C., Tedder, M., Corneil, D.: Practical and efficient split decomposition via graph-labelled trees. *Algorithmica* **69**(4), 789–843 (2014)
15. Hsu, W.L.:  $O(M \cdot N)$  algorithms for the recognition and isomorphism problems on circular-arc graphs. *SIAM J. Comput.* **24**(3), 411–439 (1995)
16. Lin, M.C., Souignac, F.J., Szwarzfiter, J.L.: A simple linear time algorithm for the isomorphism problem on proper circular-arc graphs. In: Gudmundsson, J. (ed.) *SWAT 2008*. LNCS, vol. 5124, pp. 355–366. Springer, Heidelberg (2008). [https://doi.org/10.1007/978-3-540-69903-3\\_32](https://doi.org/10.1007/978-3-540-69903-3_32)
17. Naji, W.: Graphes de cordes: une caracterisation et ses applications, Ph. D. thesis, l'Université Scientifique et Médicale de Grenoble (1985)
18. Oum, S.: Rank-width and vertex-minors. *J. Comb. Theory Ser. B* **95**(1), 79–100 (2005)
19. Shiloach, Y.: Fast canonization of circular strings. *J. Algorithms* **2**(2), 107–121 (1981)
20. Spinrad, J.P.: Recognition of circle graphs. *J. Algorithms* **16**(2), 264–282 (1994)
21. Spinrad, J.P.: Efficient graph representations. *Field Institute Monographs* (2003)





# The Exact Subset MultiCover Problem

Emile Benoist<sup>(✉)</sup>, Guillaume Fertin<sup>(iD)</sup>, and Géraldine Jean<sup>(iD)</sup>

Nantes Université, CNRS, LS2N, UMR 6004, 44000 Nantes, France  
{emile.benoist,guillaume.fertin,geraldine.jean}@univ-nantes.fr

**Abstract.** In this paper, we study the EXACT SUBSET MULTICOVER problem (or ESM), which can be seen as an extension of the well-known SET COVER problem. Let  $(\mathcal{U}, f)$  be a multiset built from set  $\mathcal{U} = \{e_1, e_2, \dots, e_m\}$  and function  $f : \mathcal{U} \rightarrow \mathbb{N}^*$ . ESM is defined as follows: given  $(\mathcal{U}, f)$  and a collection  $\mathcal{S} = \{S_1, S_2, \dots, S_n\}$  of  $n$  subsets of  $\mathcal{U}$ , is it possible to find a multiset  $(\mathcal{S}', g)$  with  $\mathcal{S}' = \{S'_1, S'_2, \dots, S'_n\}$  and  $g : \mathcal{S}' \rightarrow \mathbb{N}$ , such that (i)  $S'_i \subseteq S_i$  for every  $1 \leq i \leq n$ , and (ii) each element of  $\mathcal{U}$  appears as many times in  $(\mathcal{U}, f)$  as in  $(\mathcal{S}', g)$ ? We study this problem under an algorithmic viewpoint and provide diverse complexity results such as polynomial cases, NP-hardness proofs and FPT algorithms. We also study two variants of ESM: (i) EXCLUSIVE EXACT SUBSET MULTICOVER (EESM), which asks that each element of  $\mathcal{U}$  appears in exactly one subset  $S'_i$  of  $\mathcal{S}'$ ; (ii) MAXIMUM EXCLUSIVE EXACT SUBSET MULTICOVER (MAX-EESM), an optimisation version of EESM, which asks that a maximum number of elements of  $\mathcal{U}$  appear in exactly one subset  $S'_i$  of  $\mathcal{S}'$ . For both variants, we provide several complexity results; in particular we present a 2-approximation algorithm for MAX-EESM, that we prove to be tight.

## 1 Introduction

The SET COVER problem [6] is a well-known optimization problem consisting in covering a set  $\mathcal{U} = \{e_1, e_2, \dots, e_m\}$  of elements (called the *universe*) using a minimum number of subsets of that universe (called *covering sets*), taken from a collection  $\mathcal{S} = \{S_1, S_2, \dots, S_n\}$ . The SET COVER problem can be generalized in many ways. For instance, in the WEIGHTED SET COVER, a cost is assigned to each covering set, and the goal is to cover  $\mathcal{U}$  while minimizing the total cost. Another variant, called SET MULTICOVER [4], adds a non-negative demand for each element in  $\mathcal{U}$  and requires that each of these elements have to be covered at least as much as requested – in that case, each of the covering sets can be used several times. The latter problem can be further extended to the case where the  $S_i$ s are covering *multisets* instead of sets: the problem is then called MULTISSET MULTICOVER [4]. When the number of times a covering set (resp. multiset) can be chosen is limited, we are in presence of the (MULTI)SET MULTICOVER WITH MULTIPLICITY CONSTRAINTS [4]. It is also possible to limit the *capacity*

---

E. Benoist et al.—supported by the French National Research Agency (ANR-18-CE45-004), ANR DeepProt.

of each covering set. For example, if the covering set  $S_1 = \{e_1, e_2, e_3\}$  is assigned a capacity of 1, it can only cover one of its three elements; however, it can be used several times, e.g. one copy of  $S_1$  can cover  $e_1$ , while a second copy can cover  $e_3$ ; in that case, the associated cost is twice the weight of  $S_1$ . The latter problem is called the CAPACITATED SET COVER problem [1].

In this paper, we introduce and study yet another variant of the SET COVER problem that we call the EXACT SUBSET MULTICOVER problem (ESM). This problem has biological motivations, that will be briefly explained later. Before that, we formally define ESM, illustrate it on an example, and discuss how it relates to and differs from the above mentioned problems.

EXACT SUBSET MULTICOVER (ESM)

**Instance :** A multiset  $(\mathcal{U}, f)$  of  $m$  elements with  $\mathcal{U} = \{e_1, e_2, \dots, e_m\}$  and  $f : \mathcal{U} \rightarrow \mathbb{N}^*$ . A collection  $\mathcal{S} = \{S_1, S_2, \dots, S_n\}$  of  $n$  subsets of  $\mathcal{U}$ .

**Question :** Does there exist a multiset  $(\mathcal{S}', g)$  where  $\mathcal{S}' = \{S'_1, S'_2, \dots, S'_n\}$  is such that  $S'_i \subseteq S_i \forall 1 \leq i \leq n$ , and  $g : \mathcal{S}' \rightarrow \mathbb{N}$  is such that  $\sum_{i \in \mathcal{E}_j} g(S'_i) = f(e_j) \forall 1 \leq j \leq m$  with  $\mathcal{E}_j = \{1 \leq i \leq n \mid e_j \in S'_i\}$  ?

We call *covering sets* (resp. *covering subsets*) the elements of  $\mathcal{S}$  (resp.  $\mathcal{S}'$ ). For an element  $e_j \in \mathcal{U}$ ,  $\mathcal{E}_j$  is the set of indices of the covering subsets containing  $e_j$ . We say that  $e_j$  is *exactly covered* if, in the multiset  $(\mathcal{S}', g)$ , the sum of the multiplicities of the covering subsets that contain  $e_j$  is equal to  $f(e_j)$ . We say that  $e_j$  is *covered*  $t \geq 1$  times (or simply *covered*) by a covering subset  $S'_i$  if  $e_j$  belong to  $S'_i$  and  $g(S'_i) = t$ . Note that in any solution of ESM, all elements of  $\mathcal{U}$  must be exactly covered.

An example is provided in Fig. 1, with  $\mathcal{U} = \{e_1, e_2, \dots, e_6\}$  (thus  $m = 6$ ) and the multiplicities in  $\mathcal{U}$  are respectively 3, 10, 4, 6, 8, 3 (e.g.  $f(e_2) = 10$ );  $\mathcal{S} = \{S_1, S_2, S_3, S_4\}$  (thus  $n = 4$ ) with  $S_1 = \{e_1, e_2, e_4, e_5\}$ ,  $S_2 = \{e_2, e_3, e_5, e_6\}$ ,  $S_3 = \{e_1, e_2, e_3, e_4\}$  and  $S_4 = \{e_2, e_3, e_4, e_6\}$ .

As can be seen in Fig. 1(right),  $((\mathcal{U}, f), \mathcal{S})$  is a YES-instance for ESM. First, for each  $1 \leq i \leq 4$ , we have  $S'_i \subseteq S_i$ . Then, by function  $g$  (rightmost column), we have that each element  $e_j \in \mathcal{U}$ ,  $1 \leq j \leq 6$ , is exactly covered. For example,  $e_2$  belongs to  $S'_1, S'_3$  and  $S'_4$  ( $\mathcal{E}_2 = \{1, 3, 4\}$ ), and function  $g$  applied to these covering subsets respectively returns 3, 4 and 3, for a total of  $10 = f(e_2)$ .

ESM has similarities to the SET MULTICOVER problem, but differs from it in two ways: first, in ESM each element  $e_j$  must be exactly covered while an element must be covered *at least* a certain number of times in SET MULTICOVER. Second, in ESM, a covering set  $S_i$  is allowed to only cover a subset of its elements (called the covering subset  $S'_i$ ). In that sense, ESM has the same flavor as the CAPACITATED SET COVER problem; however, it differs from it, as in ESM, the size of each covering subset is not constrained. Moreover, when a given covering set  $S_i$  is used several times in ESM, the same covering subset  $S'_i$  must be used every time, a constraint that is not present in CAPACITATED SET COVER.

The ESM problem is motivated by proteomics, and more precisely by the *protein inference* problem. The goal is to infer the protein sequences that are

	Universe $\mathcal{U}$					
Collection $\mathcal{S}$	$e_1$	$e_2$	$e_3$	$e_4$	$e_5$	$e_6$
$S_1$	<b>•</b>	<b>•</b>		<b>•</b>	<b>•</b>	
$S_2$		<b>•</b>	<b>•</b>		<b>•</b>	<b>•</b>
$S_3$	<b>•</b>	<b>•</b>	<b>•</b>	<b>•</b>		
$S_4$		<b>•</b>	<b>•</b>			<b>•</b>
$f \rightarrow$	3	10	4	6	8	3

	Universe $\mathcal{U}$						
$S'$	$e_1$	$e_2$	$e_3$	$e_4$	$e_5$	$e_6$	$g$ ↓
$S'_1$	<b>•</b>	<b>•</b>		<b>•</b>			3
$S'_2$					<b>•</b>		8
$S'_3$		<b>•</b>	<b>•</b>				4
$S'_4$		<b>•</b>		<b>•</b>		<b>•</b>	3
$f \rightarrow$	3	10	4	6	8	3	

**Fig. 1.** (Left) An instance  $((\mathcal{U}, f), \mathcal{S})$  of ESM, where dots in bold show which elements belong to which covering sets, and where the multiplicity function  $f$  of each element in  $\mathcal{U}$  is provided in the bottom row. (Right) This instance is a YES-instance for ESM, and a solution is provided. The last column corresponds to the multiplicity function  $g$  of the covering subsets  $S'_i$ s in the multiset  $(S', g)$ .

present in a biological sample, using the information provided by mass spectrometry (MS/MS). In a nutshell, each protein is cut in smaller pieces called *peptides*, before entering the mass spectrometer. The mass spectrometer then outputs a series of *spectra*, each ideally representing a peptide. Once a spectrum is associated to a peptide (through some dedicated algorithm), we obtain a multiset of peptides corresponding to the initial sample, and the aim is to decide which set of proteins (provided from a databank) corresponds with the peptides at hand (see e.g. [3], or Chap. 16 of [5], for a detailed description). In our setting, each element of  $\mathcal{U}$  represents a peptide,  $f$  is the multiplicity function of each peptide, while  $\mathcal{S}$  is the set of proteins of the databank. The ESM problem then consists in inferring the proteins from  $\mathcal{S}$  that are present in the sample, together with their abundance, given the information provided by the multiset  $(\mathcal{U}, f)$  of peptides.

We will also consider a constrained version of ESM, in which we ask that each element  $e_j$  belongs to exactly *one* covering subset  $S'_{k_j}$  (in which case we say that  $S'_{k_j}$  exactly covers  $e_j$ ). We call this version EXCLUSIVE EXACT SUBSET MULTICOVER (EESM).

**EXCLUSIVE EXACT SUBSET MULTICOVER**

**Instance :** A multiset  $(\mathcal{U}, f)$  of  $m$  elements with  $\mathcal{U} = \{e_1, e_2, \dots, e_m\}$  and  $f : \mathcal{U} \rightarrow \mathbb{N}^*$ . A collection  $\mathcal{S} = \{S_1, S_2, \dots, S_n\}$  of  $n$  subsets of  $\mathcal{U}$ .

**Question :** Does there exist a multiset  $(S', g)$  where  $S' = \{S'_1, S'_2, \dots, S'_n\}$  such that  $S'_i \subseteq S_i \forall 1 \leq i \leq n$  and  $\mathcal{E}_j = \{k_j\} \forall 1 \leq j \leq m$ , and  $g : S' \rightarrow \mathbb{N}$  such that  $g(S'_{k_j}) = f(e_j) \forall 1 \leq k_j \leq n$  ?

Note that EESM is a variant of ESM with additional constraints on the solution, hence, any YES-instance for EESM is also a YES-instance for ESM, and any NO-instance for ESM is also a NO-instance for EESM.

**Table 1.** Complexity results for ESM, EESM and MAX-EESM. Parameters  $m$  and  $n$  respectively denote the number of covering sets in  $\mathcal{S}$  and the number of elements in  $\mathcal{U}$ .

Problem	Complexity	FPT	Approximability
ESM	NP-hard (Thms 1 and 2)	wrt $m$ ; wrt $n$ (Props 2 and 3)	n/a
EESM	NP-hard (Thm 4)	wrt $m$ ; wrt $n$ (Prop. 6)	n/a
MAX-EESM	NP-hard (from EESM)	wrt $m$	ratio 2 (Thm 5)

In the following, it will be convenient to partition the different multiplicities in  $(\mathcal{U}, f)$  into *groups*, where each element having the same multiplicity belongs to the same group. For example, if  $\mathcal{U} = \{e_1, e_2, e_3\}$  and the multiplicities given by  $f$  are resp. 5, 7 and 5, then, we have two groups:  $\{e_1, e_3\}$  and  $\{e_2\}$ . We call  $n_{\mathbf{g}}$  the total number of groups for a given multiset  $(\mathcal{U}, f)$ . It can be seen that, for any YES-instance for EESM, in any solution, a covering subset  $S'_i$  can only contain elements belonging to the same group. If not, this means that some  $S'_i$  covers  $g(S'_i)$  times two elements having different multiplicities by  $f$ , which implies that at least one of these two elements is not exactly covered, a contradiction. As an illustration, the instance depicted in Fig. 1 is a NO-instance for EESM, simply because there are 5 groups and only 4 covering sets in  $\mathcal{S}$ .

In this paper, we also consider a third problem that we call MAXIMUM EXCLUSIVE EXACT SUBSET MULTICOVER (MAX-EESM), which can be seen as the maximization version of EESM: the goal is to find a maximum cardinality subset  $\mathcal{U}'$  of  $\mathcal{U}$  such that  $((\mathcal{U}', f), \mathcal{S})$  is a YES-instance for EESM.

#### MAXIMUM EXCLUSIVE EXACT SUBSET MULTICOVER

**Instance :** A multiset  $(\mathcal{U}, f)$  of  $m$  elements with  $\mathcal{U} = \{e_1, e_2, \dots, e_m\}$  and  $f : \mathcal{U} \rightarrow \mathbb{N}^*$ . A collection  $\mathcal{S} = \{S_1, S_2, \dots, S_n\}$  of  $n$  subsets of  $\mathcal{U}$ .

**Solution :** A multiset  $(\mathcal{S}', g)$  where  $\mathcal{S}' = \{S'_1, S'_2, \dots, S'_n\}$  is such that  $S'_i \subseteq S_i \forall 1 \leq i \leq n$  and  $\mathcal{E}_j = \{k_j\} \forall 1 \leq j \leq m$ , and  $g : \mathcal{S}' \rightarrow \mathbb{N}$ .

**Measure :**  $|\mathcal{U}'|$  where  $\mathcal{U}' = \{e_j \in \mathcal{U} \mid g(S'_{k_j}) = f(e_j)\}$ .

We refer to Fig. 1 for an illustration: MAX-EESM will return a subset  $\mathcal{U}'$  of  $\mathcal{U}$  with  $|\mathcal{U}'| = 4$ . In other words, it is necessary to remove 2 elements from  $\mathcal{U}$  to satisfy EESM: indeed, (i) group  $\{e_1, e_6\}$  with multiplicity 3 needs to be exactly covered by two different covering subsets, as no covering set simultaneously contains  $e_1$  and  $e_6$ , and (ii) there are 5 groups and 4 covering sets in  $\mathcal{S}$ . Moreover,  $((\mathcal{U}', f), \mathcal{S})$  with  $\mathcal{U}' = \{e_1, e_2, e_3, e_4\}$  is a YES-instance for EESM (simply take  $S'_i$  to exactly cover  $e_i$  for  $1 \leq i \leq 4$ ).

The present paper aims at studying ESM, EESM and MAX-EESM from an algorithmic viewpoint (see resp. Sects. 2, 3 and 4). Most of our results are summarized in Table 1. Due to space constraints, some of the proofs are omitted. They will be available in the journal version of this paper.

## 2 The Exact Subset MultiCover Problem

In this section, we focus on the ESM problem. We study its computational complexity, first depending on the number  $n_g$  of groups in the instance, which allows us to draw the tractability border based on that value (Proposition 1 and Theorem 1); then, depending on the contents of  $\mathcal{S}$  (Theorem 2). We then prove that ESM is fixed-parameterized tractable with respect to  $m = |\mathcal{U}|$  (Proposition 2) and with respect to  $n = |\mathcal{S}|$  (Proposition 3).

**Proposition 1.** *ESM is in P when  $n_g = 1$  (i.e., when  $f$  is a constant function).*

**Theorem 1.** *ESM is NP-hard even if (i) every covering set  $S_i$  is of cardinality at most 3; (ii) every element of  $\mathcal{U}$  is present in at most 3 covering sets; and (iii) the number  $n_g$  of groups is equal to 2.*

*Proof.* The proof is by reduction from 3-SAT-3, a constrained variant of SAT in which every clause contains at most 3 literals, and every variable appears at most 3 times in the formula. It has been shown that 3-SAT-3 is NP-hard [7].

Take any instance  $\Phi = C_1 \wedge C_2 \wedge \dots \wedge C_m$  of 3-SAT-3, where each clause  $C_j$  of  $\Phi$ ,  $1 \leq j \leq m$ , is built from boolean variables taken from  $X = \{x_1, x_2, \dots, x_n\}$ . The instance of ESM we build from  $\Phi$  is as follows: first, the set  $\mathcal{U}$  is the union of two sets  $\mathcal{U}_{cl}$  and  $\mathcal{U}_{neg}$ . More precisely,  $\mathcal{U}_{cl} = \{e_1, e_2, \dots, e_m\}$ , i.e.  $\mathcal{U}_{cl}$  contains as many elements as there are clauses in  $\Phi$ . Moreover,  $f(e_i) = 1 \forall e_i \in \mathcal{U}_{cl}$ . The second set  $\mathcal{U}_{neg}$  contains elements of the form  $e_{i,j}$ , for all  $i, j$  such that clause  $C_j$  contains the negative literal  $\bar{x}_i$ . For all elements  $e_{i,j}$  from  $\mathcal{U}_{neg}$ , we set  $f(e_{i,j}) = 3$ .

Now, the collection  $\mathcal{S}$  is also the union of two sets, namely  $\mathcal{S}_{var} \cup \mathcal{S}_{neg}$ . The first set,  $\mathcal{S}_{var} = \{S_1, S_2, \dots, S_n\}$  contains as many covering sets as there are variables in  $X$ . The second set,  $\mathcal{S}_{neg}$ , contains covering sets denoted  $S_{i,j}$ , for all  $i$  and  $j$  such that clause  $C_j$  contains the negative literal  $\bar{x}_i$  (if an element  $e_{i,j}$  belongs to  $\mathcal{U}_{neg}$ , then a covering set  $S_{i,j}$  belongs to  $\mathcal{S}_{neg}$ ). It now remains to describe the contents of each covering set  $S_i$  and  $S_{i,j}$ : for any  $1 \leq j \leq m$ , if clause  $C_j$  contains the positive literal  $x_i$  (resp. the negative literal  $\bar{x}_i$ ), then  $e_j$  belongs to  $S_i$  (resp. to  $S_{i,j}$ ). Moreover, if clause  $C_j$  contains the negative literal  $\bar{x}_i$ , then  $e_{i,j}$  belongs to both  $S_i$  and  $S_{i,j}$  – see Fig. 2 for an illustration.

The ESM instance we have built satisfies the constraints listed in the statement of Theorem 1. First, each element belongs to at most 3 covering sets of  $\mathcal{S}$ : each element  $e_j$  appears in at most 3 covering sets, since  $e_j$  represents a clause  $C_j$ , which is of size at most 3, while each element  $e_{i,j}$  appears in exactly two covering sets, namely  $S_i$  and  $S_{i,j}$ . Second, any of the covering sets we build contains at most 3 elements: a covering set  $S_i$  contains one element per occurrence of variable  $x_i$  in  $\Phi$  (thus at most 3 by definition of 3-SAT-3), while a covering set  $S_{i,j}$  contains exactly 2 elements by construction. Finally, by construction, each element appears 1 time or 3 times in the multiset  $(\mathcal{U}, f)$ , thus  $n_g = 2$ .

We now show that  $\Phi$  is a YES-instance for 3-SAT-3 iff  $((\mathcal{U}, f), \mathcal{S})$  is a YES-instance for ESM.

( $\Rightarrow$ ) Let  $\Phi$  be a YES-instance of 3-SAT-3; thus every clause in  $\Phi$  is satisfied. For each variable  $x_i \in X$ , if  $x_i = \text{True}$ , then we set  $S'_i = S_i \cap \mathcal{U}_{cl}$  with  $g(S'_i) = 1$ .

Collection $\mathcal{S}$	Universe $\mathcal{U}$						
	$e_1$	$e_2$	$e_3$	$e_4$	$e_{5,2}$	$e_{2,3}$	$e_{4,4}$
$S_1$	•	•					
$S_2$	•					•	
$S_3$	•		•	•			
$S_4$		•					•
$S_5$			•		•		
$S_{5,2}$		•			•		
$S_{2,3}$			•			•	
$S_{4,4}$				•			•
$f \rightarrow$	1	1	1	1	3	3	3

**Fig. 2.** Illustration of our reduction from 3-SAT-3, with  $\Phi = (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_4 \vee \overline{x_5}) \wedge (\overline{x_2} \vee x_3 \vee x_5) \wedge (x_3 \vee \overline{x_4})$ ,  $n = 5$  and  $m = 4$ . Clause  $C_2 = (x_1 \vee x_4 \vee \overline{x_5})$  corresponds to element  $e_2$  (with  $f(e_2) = 1$ ). Element  $e_2$  is present in covering sets  $S_1$  and  $S_4$  since  $C_2$  contains the two positive literals  $x_1$  and  $x_4$ . Clause  $C_2$  contains one negative literal,  $\overline{x_5}$ , hence the existence of element  $e_{5,2}$  in covering set  $S_{5,2}$  (with  $f(e_{5,2}) = 3$ ).

If, on the contrary,  $x_i = \text{False}$ , then we set  $S'_i = S_i \cap \mathcal{U}_{neg}$  with  $g(S'_i) = 3$ . In case an element  $e_j$  from  $\mathcal{U}_{cl}$  is simultaneously present in several distinct  $S'_i$ s, we remove  $e_j$  from all  $S'_i$ s containing it except one chosen arbitrarily (the covering subsets  $S'_i$ s from which  $e_j$  has been removed remain subsets of  $S_i$ ). Now, if for some  $j$ ,  $e_j$  is not present in any  $S'_i$ , this means there exists an  $i'$  such that (i)  $e_j$  is in  $S_{i',j}$ , and (ii)  $\overline{x_{i'}} = \text{True}$ , since  $C_j$  is satisfied. In that case, we set  $S'_{i',j} = \{e_j\}$  and  $g(S'_{i',j}) = 1$ . Finally, if, for some  $i$  and  $j$ ,  $e_{i,j}$  is not present in  $S_i$ , we set  $S'_{i,j} = \{e_{i,j}\}$  and  $g(S'_{i,j}) = 3$ . Having done that, we have exactly covered every element from  $\mathcal{U}$ . It just remains to show that we introduced no inconsistency on the way. First, for a fixed  $i$ ,  $g(S'_i)$  cannot be equal to 1 and 3 at the same time, by construction. It thus remains to show that a given  $g(S'_{i,j})$  cannot be equal to 1 and 3 at the same time. For this, recall that  $e_{i,j} \in S'_{i,j}$  (and  $g(S'_{i,j}) = 3$ ) only when  $e_{i,j} \notin S'_i$ , i.e. when  $\overline{x_i}$  is present in  $C_j$  with  $x_i = \text{True}$ . On the other hand, as mentioned above,  $e_j \in S'_{i,j}$  (and  $g(S'_{i,j}) = 1$ ) only when  $x_i = \text{False}$ . Hence, no inconsistency occurs, and our construction is thus a YES-instance for ESM.

( $\Leftarrow$ ) Suppose the constructed instance  $((\mathcal{U}, f), \mathcal{S})$  of ESM is a YES-instance. For each covering set  $S_i$  in  $\mathcal{S}_{var}$ , if  $f(S'_i) \leq 1$  or if  $S'_i = \emptyset$ , then we set  $x_i$  to **True**. Otherwise (i.e. if  $S'_i$  contains only and at least one element of  $\mathcal{U}_{neg}$  and if  $f(S'_i) \geq 2$ ), we set  $x_i$  to **False**. Note first that an element  $e_j$  from  $\mathcal{U}_{cl}$  cannot be simultaneously covered by several distinct covering subsets in  $\mathcal{S}'$ , since  $f(e_j) = 1$ . Let us now show that the above described assignment satisfies the formula  $\Phi$  from the initial 3-SAT-3 problem. Consider an element  $e_j$ : if it is covered by a subset  $S'_i$ , this is necessarily with  $g(S'_i) = 1$ , thus  $x_i = \text{True}$ , which satisfies  $C_j$  since  $C_j$  contains literal  $x_i$  by construction. Suppose now  $e_j$  is covered by a subset  $S'_{i,j}$ , thus with  $g(S'_{i,j}) = 1$ . Then, we know that  $e_{i,j}$  is either covered one time by  $S'_{i,j}$  (if  $e_{i,j} \in S'_{i,j}$ ), or not covered by it (if  $e_{i,j} \notin S'_{i,j}$ ). In any case, this proves that

$e_{i,j}$  must belong to  $S'_i$  with  $g(S'_i) \in \{2, 3\}$ , since  $e_{i,j}$  is only present in  $S_{i,j}$  and  $S_i$ . Hence, we conclude that  $x_i = \text{False}$  (since  $g(S'_i) \geq 2$ ,  $e_{i,j} \in S'_i$  and  $S'_i$  cannot contain any element from  $\mathcal{U}_{cl}$  because  $g(S'_i) \geq 2$ ), which satisfies  $C_j$  since the existence of  $e_{i,j}$  implies the presence of literal  $\bar{x}_i$  in  $C_j$ .

**Corollary 1.** *Unless ETH fails, there exists a  $\delta > 0$  such that ESM cannot be solved in  $O(2^{\delta \cdot \max(n,m)})$ .*

**Theorem 2.** *ESM is NP-hard, even if every covering set  $S_i$  is of cardinality 2.*

*Proof.* The proof is by reduction from the NP-hard PARTITION problem [2,6]: given a multiset  $\mathcal{Q} = \{q_1, q_2, \dots, q_n\}$  of positive integers such that  $C = \sum_{i=1}^n q_i$  is even, is it possible to partition  $\mathcal{Q}$  in two subsets  $\mathcal{Q}_1$  and  $\mathcal{Q}_2$  such that  $\sum_{q_i \in \mathcal{Q}_1} = \sum_{q_i \in \mathcal{Q}_2} = \frac{C}{2}$ ? Let  $\mathcal{Q} = \{q_1, q_2, \dots, q_n\}$  be any instance of PARTITION. We construct an instance of ESM as follows: first, let  $\mathcal{U} = \{e_1, e_2, \dots, e_n, e_{n+1}\}$  with  $f(e_i) = q_i$  for every  $1 \leq i \leq n$ , and  $f(e_{n+1}) = \frac{C}{2}$ ; let also  $\mathcal{S} = \{S_1, S_2, \dots, S_n\}$  where  $S_i = \{e_i, e_{n+1}\}$ ,  $1 \leq i \leq n$ . Clearly, in this instance of ESM, every covering set  $S_i$  contains exactly two elements. We now show that  $\mathcal{Q}$  is a YES-instance for PARTITION iff  $((\mathcal{U}, f), \mathcal{S})$  is a YES-instance for ESM.

( $\Rightarrow$ ) Suppose  $\mathcal{Q}$  is a YES-instance for PARTITION. Thus, there exists a partition  $\{\mathcal{Q}_1, \mathcal{Q}_2\}$  of  $\mathcal{Q}$  such that  $\sum_{q_i \in \mathcal{Q}_1} = \sum_{q_i \in \mathcal{Q}_2} = \frac{C}{2}$ . For each  $q_i \in \mathcal{Q}_1$ , we let  $S'_i = \{e_i, e_{n+1}\}$  and  $g(S'_i) = q_i$ . For each  $q_i \in \mathcal{Q}_2$ , we let  $S'_i = \{e_i\}$  and  $g(S'_i) = q_i$ . We can see that each element  $e_i$ ,  $1 \leq i \leq n$ , is exactly covered by  $S'_i$ , whereas  $e_{n+1}$  is covered by some covering subsets of  $\mathcal{S}'$ , namely those whose indices correspond to the indices of the  $q_i$ s that are in  $\mathcal{Q}_1$ . Since  $\sum_{q_i \in \mathcal{Q}_1} q_i = \frac{C}{2}$ ,  $e_{n+1}$  is altogether exactly covered. Thus the solution we provide is valid, and  $((\mathcal{U}, f), \mathcal{S})$  is a YES-instance for ESM.

( $\Leftarrow$ ) Suppose  $((\mathcal{U}, f), \mathcal{S})$  is a YES-instance for ESM. Hence, every element is exactly covered, and in particular elements  $\{e_1, e_2, \dots, e_n\}$ . Since each  $e_i$ ,  $1 \leq i \leq n$ , appears only in  $S_i$ , each  $S'_i$  contains  $e_i$  with  $g(S'_i) = q_i$ . This implies that some covering subsets of  $\mathcal{S}'$  collectively exactly cover  $e_{n+1}$ . Since  $g(S'_i) = q_i$  for all  $1 \leq i \leq n$ , we conclude that  $\frac{C}{2}$  is the sum of the  $g(S'_i)$ s for which  $S'_i$  contains  $e_{n+1}$ . Hence, a solution for PARTITION can be inferred from the observed solution of ESM: if a covering subset  $S'_i$  contains  $e_{n+1}$ , assign  $q_i$  to  $\mathcal{Q}_1$ , otherwise assign  $q_i$  to  $\mathcal{Q}_2$ . We have that  $\{\mathcal{Q}_1, \mathcal{Q}_2\}$  is a partition of  $\mathcal{Q}$ . Moreover, the above argument shows that  $\sum_{q_i \in \mathcal{Q}_1} q_i = \frac{C}{2}$ , and thus  $\sum_{q_i \in \mathcal{Q}_2} q_i = \frac{C}{2}$  as well.

The two following results show that ESM is FPT with respect to  $m$ , the size of the universe  $\mathcal{U}$ ; and FPT with respect to  $n$ , the size of the collection  $\mathcal{S}$ .

**Proposition 2.** *ESM is FPT with respect to parameter  $m = |\mathcal{U}|$ .*

*Proof.* First, given that no two covering sets of  $\mathcal{S}$  are identical, and given that all covering sets of  $\mathcal{S}$  are built from  $\mathcal{U}$ , we have that  $|\mathcal{S}| \leq 2^{|\mathcal{U}|}$ , i.e.  $n \leq 2^m$ . Then, consider the following algorithm : (i) generate all possible  $\mathcal{S}' = \{S'_1, S'_2, \dots, S'_n\}$  such that  $S'_i \subseteq S_i \forall 1 \leq i \leq n$ ; then, for each generated  $\mathcal{S}'$ , (ii) solve the resulting problem (see below). Note that it may not be necessary to generate all possible

$\mathcal{S}'$ , as the algorithm stops when one solution is found. In the worst case, the size of  $\mathcal{S}'$  is  $O(2^m)$  given that  $|\mathcal{S}'| = |\mathcal{S}|$ . Moreover, for each covering subset  $S'_i$ , we have  $S'_i \in \mathcal{P}(S_i)$  where  $\mathcal{P}(S_i)$  is the power set of  $S_i$ . Since  $|S_i| \leq m$ , then,  $|\mathcal{P}(S_i)| \leq 2^m$ , and the number of possibilities in (i) does not exceed  $2^{m2^m}$ .

Given any  $\mathcal{S}'$ , solving the resulting problem as mentioned in (ii) above consists in finding the multiplicity function  $g$ . This problem can be modeled as a system of linear equations where the variables are the  $g(S'_i)$ s,  $1 \leq i \leq n$ . For example, if  $e_1$  belongs to  $S'_1, S'_3$  and  $S'_4$ , we create the following equation:  $f(e_1) = g(S'_1) + g(S'_3) + g(S'_4)$ . The number of variables (resp. constraints) of such a system is at most  $2^m$  (resp.  $m$ ). This system is solvable in a time that only depends on the number of variables and equations. Thus the total complexity of our algorithm depends only on  $m$ , which proves the result.

**Proposition 3.** *ESM is FPT with respect to parameter  $n = |\mathcal{S}|$ .*

### 3 The Exclusive Exact Subset MultiCover Problem

In this section, we focus on the EESM problem, in which every element of  $\mathcal{U}$  must be exactly covered by a *single* covering subset of  $\mathcal{S}'$ . We start with two positive results, before discussing NP-hardness and fixed-parameterized tractability.

**Proposition 4.** *EESM is in P (i) when  $n_g = 1$ ; (ii) when  $n_g = |\mathcal{U}|$  (i.e., each element  $e$  of  $\mathcal{U}$  has distinct multiplicity); and (iii) when  $n_g = |\mathcal{S}|$ .*

**Theorem 3.** *EESM is in P when every element of  $\mathcal{U}$  belongs to at most two covering sets of  $\mathcal{S}$ .*

*Proof.* First note that we can always assume that each element belongs to *exactly* two covering sets of  $\mathcal{S}$ . Indeed, when an element  $e_j$  belongs to exactly one covering set  $S_i$ , then necessarily, in any YES-instance, the chosen  $S'_i$  must exactly cover  $e_j$ , and thus the initial instance can be reduced, by removing  $S_i$  and each element  $e \in S_i$  such that  $f(e) = f(e_j)$ .

The proof is by equivalence with 2-SAT, which is known to be in P. Given any EESM instance  $((\mathcal{U}, f), \mathcal{S})$  such that every element belongs to two covering sets, we construct an instance  $\Phi$  of 2-SAT on a set  $X$  of variables defined as follows: for each covering set  $S_i \in \mathcal{S}$  and for each element  $e_j \in S_i$ , add the boolean variable  $x_{i,j}$  to  $X$ . Now we show how to build a 2-SAT formula  $\Phi$  using variables from  $X$ . We build two categories of clauses. The first one, which we call *assignment* clauses, is as follows: for each element  $e_j$  (thus present in two covering sets, say  $S_{i_1}$  and  $S_{i_2}$ ), we create the two following clauses:  $C_j^1 = (x_{i_1,j} \vee x_{i_2,j})$  and  $C_j^2 = (\overline{x_{i_1,j}} \vee \overline{x_{i_2,j}})$ . Note that the conjunction of  $C_j^1$  and  $C_j^2$  is equivalent to the XOR expression  $(x_{i_1,j} \oplus x_{i_2,j})$ : this encodes the fact that every element  $e_j$  must be exactly covered by only one covering subset (either  $S'_{i_1}$  or  $S'_{i_2}$ ). The second category, which we call *consistency* clauses, will encode the fact that a given covering subset  $S'_i$  cannot simultaneously contain two elements  $e_{j_1}$  and  $e_{j_2}$  such that  $f(e_{j_1}) \neq f(e_{j_2})$ . Thus, for each covering set  $S_i$ , and for each pair of



elements  $e_{j_1}, e_{j_2} \in S_i$  such that  $f(e_{j_1}) \neq f(e_{j_2})$ , we create the following clause:  $(\overline{x_{i,j_1}} \vee \overline{x_{i,j_2}})$ . Finally, the 2-SAT formula  $\Phi$  we construct is the conjunction of all assignment clauses and consistency clauses.

It can be seen that  $((\mathcal{U}, f), \mathcal{S})$  is a YES-instance for EESM iff  $\Phi$  is a YES-instance for 2-SAT. Indeed, suppose  $((\mathcal{U}, f), \mathcal{S})$  is a YES-instance for EESM and observe a given solution: for each element  $e_j$  exactly covered by (resp. not in)  $S'_i$ , set  $x_{i,j}$  to **True** (resp. to **False**). In that case, all assignment clauses are satisfied, as any element is exactly covered by one covering subset in  $\mathcal{S}'$ . Moreover, consistency clauses are also satisfied, since no two elements with distinct multiplicities in  $(\mathcal{U}, f)$  can belong to the same covering subset in a solution to EESM. Now, suppose  $\Phi$  is a YES-instance for 2-SAT and observe a given solution: if  $x_{i,j} = \text{True}$ , then, covering subset  $S'_i$  exactly covers element  $e_j$  (with  $e_j \in S'_i$  and  $g(S'_i) = f(e_j)$ ), otherwise, it does not contain it ( $e_j \notin S'_i$ ). This leads to a positive solution for EESM since assignment clauses ensure that each element is exactly covered by exactly one subset, while consistency clauses ensure that a covering subset does not contain elements having different multiplicities.

Although we provided in Proposition 4 and Theorem 3 classes of instances for which EESM is in P, as shown in the following theorem, the problem becomes NP-hard, even if the input instance is constrained. Note that Theorems 3 and 4 collectively draw the border between tractable and intractable instances, when considering parameter “number of covering sets to which an element belongs”.

**Theorem 4.** *EESM is NP-hard even if (i) every covering set  $S_i$  is of cardinality at most 3; (ii) every element of  $\mathcal{U}$  is present in at most 3 covering sets of  $\mathcal{S}$ ; and (iii) the number  $n_g$  of groups is equal to 2.*

*Proof.* The proof is adapted from proof of Theorem 1, and will only be briefly described here. As in proof of Theorem 1, we reduce from 3-SAT-3, and the reduction is very similar, the only difference being that  $f(e_j)$  and  $f(e_{i,j})$  do not need to be exactly 1 and 3, but just need to be different.

The forward direction is the same as in proof of Theorem 1, since the constructed solution for ESM appears to be a solution for EESM as well. The reverse direction is based on the same arguments, but appears to be simpler, due to the specificity of EESM, namely, a covering subset of  $\mathcal{S}'$  cannot contain two elements with different multiplicities.

Because the above theorem is proved similarly as in Theorem 1, the same result as Corollary 1 also applies to EESM.

**Corollary 2.** *Unless ETH fails, there exists a  $\delta > 0$  such that EESM cannot be solved  $O(2^{\delta \cdot \max(n,m)})$ .*

Since EESM is NP-hard, it is natural to ask for the existence of moderate exponential-time algorithms. We start with the following proposition – recall that  $n = |\mathcal{S}|$  is the number of covering sets, and  $n_g$  is the number of groups.

**Proposition 5.** *EESM is in XP parameterized by  $\ell = n - n_g$ .*

It can also be easily seen that EESM is FPT in  $m = |\mathcal{U}|$ . Indeed, since  $\mathcal{S}$  does not contain twice the same subsets of  $\mathcal{U}$ , we have  $|\mathcal{S}| \leq 2^m$ . Hence, only the values provided by  $f$  do not depend on  $m$ . However, by definition of EESM, these values cannot be split, and thus, only the presence/absence of an element in the covering subsets of  $\mathcal{S}'$  needs to be inferred. Hence, a solution for EESM can be computed with a time complexity that only depends on  $m$ , and thus, EESM is FPT in  $m$ . We also have the following result.

**Proposition 6.** *EESM is FPT with respect to parameter  $n = |\mathcal{S}|$ .*

*Proof.* We propose an algorithm performing a search in a bounded tree  $T$  of arity at most  $n$  and of height at most  $n$ . The way the algorithm works is the following: at each step, consider a so far uncovered element  $e$ , and guess, among the remaining covering subsets  $S'_i$ 's for which  $S_i$  contains  $e$ , which one exactly covers  $e$ . Moreover, when a covering subset  $S'_i$  is assumed to exactly cover  $e$ , we enforce *all* elements in the same group as  $e$  which also belong to  $S_i$  to be also exactly covered by  $S'_i$ . Then, we remove all exactly covered elements, as well as  $S_i$ , from the instance. The algorithm stops when an uncovered element cannot be exactly covered, or when all elements have been exactly covered. In the latter case, we have a YES-instance whose solution can be computed by backtracking. The fact that the former case always corresponds to a NO-instance derives from the following claim: any (positive) solution to EESM can be modified into a solution found by our algorithm. Indeed, if a solution  $Sol$  to EESM is not found by our algorithm, then necessarily, in  $Sol$ , there exists a group in which two elements, say  $e_1$  and  $e_2$ , are exactly covered by two different covering subsets, say  $S'_1$  and  $S'_2$ . Moreover,  $e_1$  and  $e_2$  are both present either in  $S'_1$ , or in  $S'_2$ , or both (otherwise our algorithm would have “tried” the configuration proposed by  $Sol$ ). Consequently, we can modify  $Sol$  so that both  $e_1$  and  $e_2$  are exactly covered by the same covering subset. It suffices to iterate this argument to prove the above claim, and this also proves that our algorithm is correct.

Now, since each new node in  $T$  corresponds to a new instance in which at least one covering set of  $\mathcal{S}$  has been discarded,  $T$  is of height at most  $n$ . Moreover, the number of guesses at each step, and thus the arity of  $T$ , also never exceeds  $n$ . Altogether, the size of  $T$  is thus in  $O(n^n)$ . Given that the time needed to create a new node in  $T$  is in  $O(m)$ , the overall complexity of our algorithm is in  $O(mn^n)$ , and the result follows.

## 4 The Max-EESM Problem

Clearly, MAX-EESM is NP-hard under the same conditions as EESM (see Theorem 4), since deciding whether  $\mathcal{U}' = \mathcal{U}$  is equivalent to solving EESM. Also, MAX-EESM is FPT parameterized by  $m = |\mathcal{U}|$ , for the same reasons as the ones developed in Sect. 3, since all computations depend only on  $m$ . We now show in Theorem 5 that a 2-approximation algorithm  $\mathcal{A}$  exists for MAX-EESM, and prove in Proposition 7 that the ratio 2 obtained by  $\mathcal{A}$  is asymptotically tight.

**Theorem 5.** *Max-EESM is 2-approximable.*

*Proof.* The algorithm  $\mathcal{A}$  we propose is greedy, and works as follows: at each iteration, choose among the remaining covering sets in  $\mathcal{S}$  the one, say  $S_i$ , that contains the most elements of the same group, that we name  $e_{k_1}, e_{k_2}, \dots, e_{k_j}$ . Hence, all these elements have the same multiplicity, say  $q$ . Set  $S'_i = \{e_{k_1}, e_{k_2}, \dots, e_{k_j}\}$ ,  $g(S'_i) = q$ , remove  $S_i$  as well as  $e_{k_1}, e_{k_2}, \dots, e_{k_j}$  from the instance and iterate. Algorithm  $\mathcal{A}$  stops when all elements have been exactly covered, or when no element can be further exactly covered. For simplicity, we assume that the number of iterations of  $\mathcal{A}$  is exactly  $n$ , e.g. by allowing “empty” iterations during which a covering set  $S_i$  is selected with  $S'_i = \emptyset$ . We now introduce some notations: let  $Sol_i^*$  (resp.  $Sol_i$ ) the set of elements that is exactly covered by  $S'_i$  in an optimal solution (resp. by  $\mathcal{A}$ ), and by  $x_i^*$  (resp.  $x_i$ ) the cardinality of  $Sol_i^*$  (resp.  $Sol_i$ ). For any  $1 \leq i < i' \leq n$ , we denote by  $\Delta_i^{i'} = Sol_i \cap Sol_{i'}^*$  the set of elements that are exactly covered both by covering subset  $S'_i$  (by  $\mathcal{A}$ ) and by covering subset  $S'_{i'}$  (in an optimal solution); we also let  $\delta_i^{i'} = |\Delta_i^{i'}|$ . Finally, for any  $1 \leq i \leq n$ , we denote by  $\Delta_i = Sol_i \cap (\bigcup_{i'=i+1}^n Sol_{i'}^*) = \bigcup_{i'=i+1}^n \Delta_i^{i'}$  the set of elements that are exactly covered both by covering subset  $S'_i$  by  $\mathcal{A}$  and any covering subset  $S'_{i'}$  with  $i' > i$  in an optimal solution. We finally let  $\delta_i = |\Delta_i|$ .

Let  $N^*$  be the number of exactly covered elements in an optimal solution to MAX-EESM, and  $N_{\mathcal{A}}$  the number of exactly covered elements by  $\mathcal{A}$ . Clearly,  $N^* = \sum_{i=1}^n x_i^*$  and  $N_{\mathcal{A}} = \sum_{i=1}^n x_i$ . Our goal is to show that  $N_{\mathcal{A}} \geq \frac{N^*}{2}$ , or otherwise stated  $2 \sum_{i=1}^n x_i - \sum_{i=1}^n x_i^* \geq 0$  (1). First note that for any  $i' \neq i''$ , we know that  $Sol_{i'}^* \cap Sol_{i''}^* = \emptyset$ , since in EESM, an element can only belong to a single covering subset of  $\mathcal{S}'$ . Thus, for any  $i$  and  $i' \neq i''$  with  $\min(i', i'') > i$ , we have  $\Delta_i^{i'} \cap \Delta_i^{i''} = \emptyset$ , and consequently, for any  $1 \leq i \leq n$ ,  $|\bigcup_{i'=i+1}^n \Delta_i^{i'}| = \sum_{i'=i+1}^n \delta_i^{i'}$ , which in turn gives:  $\delta_i = \sum_{i'=i+1}^n \delta_i^{i'} \forall 1 \leq i \leq n$  (2). Coming back to inequality (1), it is equivalent to:  $2 \sum_{i=1}^n x_i - \sum_{i=1}^n x_i^* + (\sum_{i=1}^n \delta_i - \sum_{i=1}^n \delta_i) \geq 0$  (3) which, by equality (2), can also be rewritten as:  $2 \sum_{i=1}^n x_i - \sum_{i=1}^n x_i^* + (\sum_{i=1}^n \sum_{i'=i+1}^n \delta_i^{i'} - \sum_{i=1}^n \delta_i) \geq 0$  (4). The left side of the above inequality can be written as  $\mathcal{I}_1 + \mathcal{I}_2$ , where  $\mathcal{I}_1 = \sum_{i=1}^n x_i - \sum_{i=1}^n \delta_i$  and  $\mathcal{I}_2 = \sum_{i=1}^n x_i + \sum_{i=1}^n \sum_{i'=i+1}^n \delta_i^{i'} - \sum_{i=1}^n x_i^*$ . Our goal is thus to show that  $\mathcal{I}_1 + \mathcal{I}_2 \geq 0$ . For this, we will separately show that  $\mathcal{I}_1 \geq 0$  and  $\mathcal{I}_2 \geq 0$ , which will allow us to conclude. First, it can be easily seen that  $\mathcal{I}_1 \geq 0$ , by noticing that  $x_i \geq \delta_i$  for any  $1 \leq i \leq n$ . Indeed, by definition, for any  $1 \leq i \leq n$ ,  $\delta_i = |Sol_i \cap (\bigcup_{i'=i+1}^n Sol_{i'}^*)|$ , and thus  $\delta_i \leq |Sol_i|$ . It suffices to recall that  $x_i = |Sol_i|$  by definition to conclude.

Now let us show that  $\mathcal{I}_2 \geq 0$ . Recall that, at any iteration  $1 \leq i \leq n$ , algorithm  $\mathcal{A}$  chooses a covering subset  $S'_i$  to exactly cover all elements belonging to the most represented group. Thus, we conclude that  $Sol_i$  contains at least as many elements as  $Sol_i^*$ , minus those which may have already been exactly covered by a covering subset  $S'_{i'}$ ,  $i' < i$ , in a previous iteration of  $\mathcal{A}$ . Hence,  $|Sol_i| \geq |Sol_i^* / \bigcup_{i'=1}^{i-1} Sol_{i'}^*| \forall 1 \leq i \leq n$  (5). Since for any two sets  $A$  and  $B$ , we have  $|A/B| = |A| - |A \cap B|$ , we know that  $|Sol_i^* / \bigcup_{i'=1}^{i-1} Sol_{i'}^*| = |Sol_i^*| - |Sol_i^* \cap \bigcup_{i'=1}^{i-1} Sol_{i'}^*|$ . Since  $x_i = |Sol_i|$  and  $x_i^* = |Sol_i^*|$ , from (5) we have:  $x_i \geq x_i^* - |Sol_i^* \cap \bigcup_{i'=1}^{i-1} Sol_{i'}^*| \forall 1 \leq i \leq n$  (6). Given that  $|Sol_i^* \cap \bigcup_{i'=1}^{i-1} Sol_{i'}^*| = |\bigcup_{i'=1}^{i-1} (Sol_i^* \cap Sol_{i'}^*)|$

$Sol_{i'}]$ , we can rewrite (6) as follows:  $x_i \geq x_i^* - |\bigcup_{i'=1}^{i-1} (Sol_i^* \cap Sol_{i'})| \forall 1 \leq i \leq n$  (7). Recall that, by definition,  $\bigcup_{i'=1}^{i-1} (Sol_i^* \cap Sol_{i'}) = \bigcup_{i'=1}^{i-1} \Delta_{i'}^i$ . We also know that all  $\Delta_{i'}^i$  are pairwise disjoint, again by definition; and since  $\delta_{i'}^i = |\Delta_{i'}^i|$ , inequality (7) becomes:  $x_i \geq x_i^* - \sum_{i'=1}^{i-1} \delta_{i'}^i$  (8). This inequality is true for any  $1 \leq i \leq n$ . Thus, if we sum it for every  $1 \leq i \leq n$ , we obtain:  $\sum_{i=1}^n x_i \geq \sum_{i=1}^n x_i^* - \sum_{i=1}^n \sum_{i'=1}^{i-1} \delta_{i'}^i$ , which we can rewrite:  $\sum_{i=1}^n x_i + \sum_{i=1}^n \sum_{i'=1}^{i-1} \delta_{i'}^i - \sum_{i=1}^n x_i^* \geq 0$  (9). We are now ready to conclude, as it can be seen that the left term of (9) is exactly  $\mathcal{I}_2$ . For this, it suffices to note that  $\sum_{i=1}^n \sum_{i'=1}^{i-1} \delta_{i'}^i$  in (9) can also be expressed as follows:  $\sum_{i=1}^n \sum_{j=i+1}^n \delta_i^j$ . Altogether, this proves  $\mathcal{I}_2 \geq 0$ . Since we proved  $\mathcal{I}_1 \geq 0$ , we have  $\mathcal{I}_1 + \mathcal{I}_2 \geq 0$ , which shows that  $N_A \geq \frac{N^*}{2}$ .

**Proposition 7.** *The approximation ratio 2 for  $\mathcal{A}$  is tight.*

## 5 Conclusion





In this paper, we have introduced the ESM problem and two of its variants, initially motivated by protein inference in mass spectrometry. For these problems, we have provided several algorithmic results, including computational complexity, parameterized complexity and approximation. Several questions remain open, for instance the following: is ESM strongly NP-hard? What is the complexity of ESM when each multiplicity in  $\mathcal{U}$  is unique? Can the 2-approximation ratio of MAX-EESM be improved, and which inapproximability ratio exists?

## References

1. Chuzhoy, J., (Seffi) Naor, J.: Covering problems with hard capacities. *SIAM J. Comput.* **36**(2), 498–515 (2006)
2. Garey, M.R., Johnson, D.S., Freeman, W.H.: *Computers and Intractability: a Guide to the Theory of NP-Completeness* (1979)
3. He, Z., Huang, T., Zhao, C., Teng, B.: Protein inference. In: Mirzaei, H., Carrasco, M. (eds.) *Modern Proteomics – Sample Preparation, Analysis and Practical Applications*. AEMB, vol. 919, pp. 237–242. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-41448-5\\_12](https://doi.org/10.1007/978-3-319-41448-5_12)
4. Hua, Q.-S., Yu, D., Lau, F.C.M., Wang, Y.: Exact Algorithms for Set Multicover and Multiset Multicover Problems. In: Dong, Y., Du, D.-Z., Ibarra, O. (eds.) *ISAAC 2009*. LNCS, vol. 5878, pp. 34–44. Springer, Heidelberg (2009). [https://doi.org/10.1007/978-3-642-10631-6\\_6](https://doi.org/10.1007/978-3-642-10631-6_6)
5. Ingvar, E., Kristian, F., Lennart, M., Svein-Ole, M.: *Computational Methods for Mass Spectrometry Proteomics*. Wiley (2008)
6. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W. (eds.) *Complexity of Computer Computations*, pp. 85–103. Plenum Press, New York. The IBM Research Symposia Series (1972)
7. Tovey, C.A.: A simplified NP-complete satisfiability problem. *Discret. Appl. Math.* **8**(1), 85–89 (1984)



# Hide a Liar: Card-Based ZKP Protocol for Usowan

Léo Robert<sup>1</sup> , Daiki Miyahara<sup>2,4</sup> , Pascal Lafourcade<sup>1</sup> ,  
and Takaaki Mizuki<sup>3,4</sup> 

<sup>1</sup> University Clermont Auvergne, LIMOS, CNRS UMR 6158, Aubière, France  
{leo.robert,pascal.lafourcade}@uca.fr

<sup>2</sup> The University of Electro-Communications, Tokyo, Japan  
miyahara@uec.ac.jp

<sup>3</sup> Cyberscience Center, Tohoku University, Sendai, Japan  
mizuki+lncs@tohoku.ac.jp

<sup>4</sup> National Institute of Advanced Industrial Science and Technology, Tokyo, Japan

**Abstract.** A Zero-Knowledge Proof (ZKP) protocol allows a participant to prove the knowledge of some secret without revealing any information about it. While such protocols are typically executed by computers, there exists a line of research proposing physical instances of ZKP protocols. Up to now, many card-based ZKP protocols for pen-and-pencil puzzles, like Sudoku, have been designed. Those games, mostly edited by Nikoli, have simple rules, yet designing them in card-based ZKP protocols is non-trivial. This is partly due to the fact that the solution should not be leaked during the protocol. In this work, we propose a card-based protocol for Usowan, a Nikoli game. In Usowan, for each room of a puzzle instance, there is exactly one piece of false information. The goal of the game is to detect this wrong data amongst the correct data and also to satisfy the other rules. Designing a card-based ZKP protocol to deal with the property of detecting a liar has never been done. In some sense, we propose a physical ZKP for hiding of a liar.

**Keywords:** Zero-knowledge proof · Pencil puzzle · Card-based cryptography · Usowan

## 1 Introduction

*Usowan* [1] is a pencil puzzle played with a rectangular grid composed of numbered cells and white cells delimited by regions (thick edges). The goal is to fill (in black) some cells:

1. The numbered cells must remain white.
2. The white cells form a connected shape.
3. The black cells cannot connect vertically or horizontally.

4. A numbered cell has the corresponding number of black cells around it (vertically or horizontally). However, each region has exactly one *liar* i.e., the number of black cells is not equal to the numbered cell.<sup>1</sup>

We depict in Fig. 1 an initial Usowan grid with its solution.

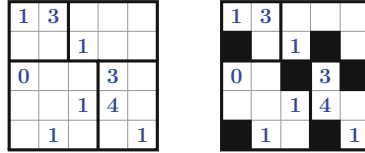


Fig. 1. Initial Usowan grid and its solution taken from [1].

Suppose that someone has found a solution for a given Usowan instance. Is it possible to design a protocol to convince anyone that he/she has the solution without revealing it? The answer can be found in the field of cryptography. Indeed, a *Zero-Knowledge Proof* (ZKP) is a process where one party can prove the knowledge of information without revealing it. A simple application to ZKP can be related to password authentication for a website; only the person with this password can access to sensitive data but it is preferable to never reveal the password.

More formally, a ZKP protocol is between two parties: a prover  $P$  who knows a solution  $s$  to a problem and a verifier  $V$  who wants to be sure that  $P$  is indeed in possession of the solution. However, no information about  $s$  should leak during the protocol (except the information recoverable without the help of the protocol). The protocol must guarantee three security properties:

- Completeness: if  $P$  knows  $s$  then  $V$  is convinced when the protocol ends.
- Soundness: if  $P$  does not have the solution, then  $V$  will detect it during the protocol.
- Zero-knowledge:  $V$  learns nothing about  $s$ .

Usually, ZKP protocols are executed by computers. We restrict ourselves by using only physical cards and envelopes. In this paper, we present a physical ZKP protocol for Usowan. While the hardness of the resolution for the underlying problem (here filling an Usowan grid) is not crucial for a physical protocol, a usual ZKP protocol needs to be based on a NP-complete problem (otherwise the verifier could compute the secret in polynomial time). Hopefully, the NP-completeness of Usowan has been proved in [13]. This result ensures that there exists a ZKP protocol.

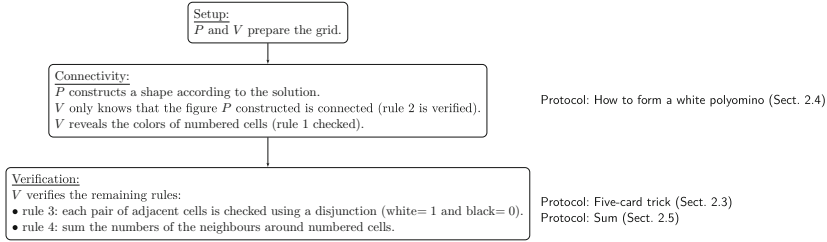
<sup>1</sup> A numbered cell whose number is four (or more) is automatically a liar. Indeed, if there are four black cells around a numbered cell, then the numbered cell cannot be connected to other white cells.

*Contributions.* We construct a physical ZKP protocol for Usowan, giving the first application to detecting if a puzzle has flaws (*i.e.*, the liar rule) while ensuring that the prover has the solution. It is the first physical ZKP protocol to prove that some information is incorrect among correct information. For this, we only use cards and envelopes. Moreover, we propose a trick that uses the rules of a Usowan grid in order to prove that exactly one piece of information is wrong in each room. For this, we use several sub-protocols to verify the rules and propose a completely novel ZKP protocol.

*Related Work.* Goldwasser *et al.* [10] proved that any NP-complete problem has its corresponding interactive ZKP protocol. Yet the generic approach has tremendous overhead leading to an impractical result. Works on implementing cryptographic protocols using physical objects are numerous, such as in [21]; or in [8] where a physical secure auction protocol was proposed. Other implementations have been studied using cards in [4, 15], polarising plates [37], polygon cards [38], a standard deck of playing cards [18], using a PEZ dispenser [2, 3], using a dial lock [19], using a 15 puzzle [20], or using a tamper-evident seals [23–25]. ZKP’s for several other puzzles have been studied such as Sudoku [30, 36], Akari [5], Takuzu [5, 16], Kakuro [5, 17], KenKen [5], Makaro [6, 35], Norinori [9], Nonogram [7, 29], Slitherlink [15], Suguru [28], Nurikabe [27], Ripple Effect [32], Numberlink [31], Bridges [33], and Cryptarithmic [12].

*Outline.* In Sect. 2, we explain how to encode a grid with some cards in order to be able to construct our ZKP protocol. We also recall the existing card-based simple protocols of the literature that we use in our construction. In Sect. 3, we present our ZKP protocol for Usowan. We give in App. D the security proofs of our protocol.

*Overview of Our Protocol.* Before detailing our protocol and existing sub-protocols involved, we present an intuition of our construction (see Fig. 2). We represent a colored cell by placing colored cards on the cell. In the connectivity phase,  $P$  and  $V$  construct a connected figure according to  $P$ ’s solution without  $V$  knowing the exact shape. Thus,  $V$  is convinced that the resulting face-down cards satisfy the rule 2 (and the rule 1 can be easily verified by just revealing face-down cards on numbered cells). Then, in the verification phase,  $V$  checks the two remaining rules. The rule 3 forces two adjacent cells to be composed of at least one white cell; this rule is verified by computing a disjunction of each possible pair. For verifying the rule 4,  $P$  and  $V$  compute the number of blacks around a given numbered cell. The result is represented as a sequence of face-down cards where the value is given by a position in the sequence. By revealing the card of position equal to the number written on the cell,  $V$  checks if the sum is equal or not (without knowing the exact value if different) to the numbered cell.



**Fig. 2.** Overview of our protocol

## 2 Preliminaries

We explain the notations and sub-protocols used in our construction; some of them are detailed in appendix while the general idea is given below. We first introduce the general framework of card-based protocols.

*Cards and Encoding.* The cards consist of clubs  $\clubsuit$  and hearts  $\heartsuit$  whose backs are identical  $\boxed{?}$ . We encode three colors {black, white, red} with the order of two cards as follows:

$$\boxed{\clubsuit}\boxed{\heartsuit} \rightarrow \text{black}, \quad \boxed{\heartsuit}\boxed{\clubsuit} \rightarrow \text{white}, \quad \boxed{\heartsuit}\boxed{\heartsuit} \rightarrow \text{red}. \quad (1)$$

We call a pair of face-down cards  $\boxed{?}\boxed{?}$  corresponding to a color according to the above encoding rule a *commitment* to the respective color. We also use the terms, a *black commitment*, a *white commitment*, and a *red commitment*. We sometimes regard black and white commitments as bit values, based on the following encoding:

$$\boxed{\clubsuit}\boxed{\heartsuit} \rightarrow 0, \quad \boxed{\heartsuit}\boxed{\clubsuit} \rightarrow 1. \quad (2)$$

For a bit  $x \in \{0, 1\}$ , if a pair of face-down cards satisfies the encoding (2), we say that it is a commitment to  $x$ , denoted by  $\underbrace{\boxed{?}\boxed{?}}_x$ .

We also define two other encodings [34]:

- $\clubsuit$ -scheme: for  $x \in \mathbb{Z}/p\mathbb{Z}$ , there are  $p$  cards composed of  $p - 1$   $\heartsuit$ s and one  $\clubsuit$  at position  $(x + 1)$  from the left. For example, 2 is represented as  $\boxed{\heartsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\heartsuit}$  in  $\mathbb{Z}/4\mathbb{Z}$ .
- $\heartsuit$ -scheme: same encoding as above but the  $\heartsuit$  and  $\clubsuit$  are reversed. For instance, 2 is represented as  $\boxed{\clubsuit}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\clubsuit}$  in  $\mathbb{Z}/4\mathbb{Z}$ .

### 2.1 Pile-shifting Shuffle [26, 38]

This shuffling action means to shuffle piles of cards *cyclically*. More formally, given  $m$  piles, each of which consists of the same number of face-down



cards, denoted by  $(\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_m)$ , applying a *pile-shifting shuffle* (denoted by  $\langle \cdot \| \dots \| \cdot \rangle$ ) results in  $(\mathbf{p}_{s+1}, \mathbf{p}_{s+2}, \dots, \mathbf{p}_{s+m})$ :

$$\left\langle \underbrace{\boxed{?}}_{\mathbf{p}_1} \parallel \underbrace{\boxed{?}}_{\mathbf{p}_2} \parallel \dots \parallel \underbrace{\boxed{?}}_{\mathbf{p}_m} \right\rangle \rightarrow \underbrace{\boxed{?}}_{\mathbf{p}_{s+1}} \underbrace{\boxed{?}}_{\mathbf{p}_{s+2}} \dots \underbrace{\boxed{?}}_{\mathbf{p}_{s+m}},$$

where  $s$  is uniformly and randomly chosen from  $\mathbb{Z}/m\mathbb{Z}$ . We can simply implement this shuffling action using physical cases that can store a pile of cards, such as boxes and envelopes. A player (or players) cyclically shuffles them manually until everyone (*i.e.*,  $P$  and  $V$ ) loses track of the offset.

### 2.2 Mizuki–Sone Copy Protocol [22]

The following protocol is used for copying commitments. We need it as a commitment can be used for several destructive<sup>2</sup> computations (here an addition). Given a commitment to  $a \in \{0, 1\}$  along with four cards  $\boxed{\clubsuit} \boxed{\heartsuit} \boxed{\clubsuit} \boxed{\heartsuit}$ , the Mizuki–Sone copy protocol [22] outputs two commitments to  $a$ . We specifically describe the protocol in Appendix A.

$$\underbrace{\boxed{?} \boxed{?}}_a \quad \boxed{\clubsuit} \boxed{\heartsuit} \boxed{\clubsuit} \boxed{\heartsuit} \quad \rightarrow \quad \underbrace{\boxed{?} \boxed{?}}_a \quad \underbrace{\boxed{?} \boxed{?}}_a.$$

### 2.3 Input-preserving Five-Card Trick [16]

This sub-protocol is used during the verification phase (see Sect. 3.3) for the lonely black rule (rule 3). Given two commitments to  $a, b \in \{0, 1\}$  based on the encoding rule (2), this sub-protocol [4, 16] reveals only the value of  $a \vee b$  as well as restores commitments to  $a$  and  $b$ :

$$\underbrace{\boxed{?} \boxed{?}}_a \quad \underbrace{\boxed{?} \boxed{?}}_b \quad \rightarrow \quad a \vee b \quad \& \quad \underbrace{\boxed{?} \boxed{?}}_a \quad \underbrace{\boxed{?} \boxed{?}}_b.$$

The original sub-protocol [4, 16] was designed for AND ( $a \wedge b$ ), but we adjust it to compute OR ( $a \vee b$ ). We give the detailed description in Appendix B.

### 2.4 How to Form a White Polyomino [27]

We introduce the idea of the generic method of [27] to perform the connectivity of colored cells without revealing any information about the resulting cells. We leave in Appendix C the details of the protocol.

First, all commitments on a grid of size  $p \times q$  are black, and  $P$  chooses a commitment to turn it white (without  $V$  knowing which cell); we use the

<sup>2</sup> This means that commitments used in the computation cannot be placed back with its initial value. A non-destructive protocol is called input-preserving (see Sect. 2.3).

chosen-pile described in Appendix C.1 for this. Then  $P$  chooses a commitment next to the previous commitment to either turn it white or leave it black;  $V$  is ensured that both commitments are neighbours (*i.e.*, two adjacent cells) using a sub-protocol described in Appendix C.2. This step is repeated  $pq - 1$  times to ensure that  $V$  does not know the number of white cells at the end of the protocol. Finally, each time a white commitment is created,  $V$  only knows that it is adjacent to another white commitment; thus  $V$  is convinced that the figure composed of white commitments is connected without knowing the number of cells.

### 2.5 Sum in $\mathbb{Z}$ [34]

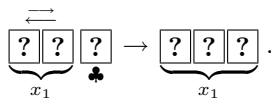
We give an overview of the protocol described in [34] for adding elements in  $\mathbb{Z}/2\mathbb{Z}$  with result in  $\mathbb{Z}$ . This protocol is needed for the liar rule 4.

Given commitments to  $x_i \in \mathbb{Z}/2\mathbb{Z}$  for  $i \in \{1, \dots, n\}$  along with one  $\spadesuit$  and one  $\heartsuit$ , the protocol produces their sum  $S = \sum_{i=1}^n x_i$  in  $\mathbb{Z}/(n+1)\mathbb{Z}$  encoded in the  $\heartsuit$ -scheme without revealing  $x_i$ . The computation is performed inductively; when starting by the two first commitments to  $x_1$  and  $x_2$ , they are transformed into  $x_1 - r$  and  $x_2 + r$  encoded in the  $\heartsuit$ -scheme and  $\spadesuit$ -scheme, respectively, for uniformly random value  $r \in \mathbb{Z}/3\mathbb{Z}$ . Then  $x_2 + r$  is revealed (no information about  $x_2$  is revealed because  $r$  is random), and  $x_1 - r$  is shifted by  $x_2 + r$  positions to encode  $(x_1 - r) + (x_2 + r) = x_1 + x_2$ . Note that this result is in  $\mathbb{Z}/(p+1)\mathbb{Z}$  (or simply  $\mathbb{Z}$  because the result is less than or equal to  $p$ ) for elements  $x_1, x_2$  in  $\mathbb{Z}/p\mathbb{Z}$ .

Let us describe the protocol. First, notice that black cells are assumed to be equal to 1 and white cells are equal to 0 (see Eqs. (1) and (2)). Two commitments to  $x_1$  and  $x_2$  (either 0 or 1) will be changed to  $x_1 + x_2$ :



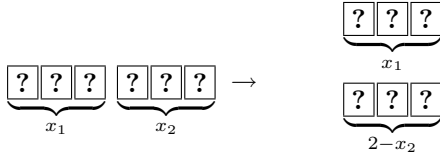
1. Swap the two cards of the commitment to  $x_1$  and add a  $\spadesuit$  face down to the right. Those three cards represent  $x_1$  in the  $\heartsuit$ -scheme in  $\mathbb{Z}/3\mathbb{Z}$ :



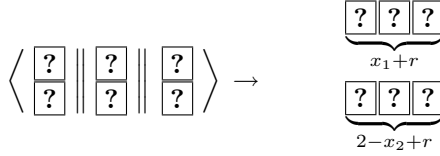
2. Add a  $\heartsuit$  on the right of the commitment to  $x_2$ . Those three cards represent  $x_2$  in the  $\spadesuit$ -scheme in  $\mathbb{Z}/3\mathbb{Z}$ :  $\underbrace{\boxed{?} \boxed{?} \boxed{?}}_{x_2} \heartsuit \rightarrow \underbrace{\boxed{?} \boxed{?} \boxed{?}}_{x_2}$ .

3. Obtain three cards representing  $x_1 + r$  and those representing  $x_2 - r$  for a uniformly random value  $r \in \mathbb{Z}/3\mathbb{Z}$  as follows.

- (a) Place in *reverse* order the three cards obtained in Step 2 below the three cards obtained in Step 1:



(b) Apply a pile shifting shuffle as follows:



For a uniformly random value  $r \in \mathbb{Z}/3\mathbb{Z}$ , we obtain three cards representing  $x_1 + r$  and those representing  $2 - x_2 + r$ .

(c) Reverse the order of the three cards representing  $2 - x_2 + r$  to obtain those representing  $x_2 - r$ :  $\underbrace{\boxed{?} \boxed{?} \boxed{?}}_{x_1+r} \underbrace{\boxed{?} \boxed{?} \boxed{?}}_{x_2-r}$ .

4. Reveal the three cards representing  $x_2 - r$ , and shift to the right the three cards representing  $x_1+r$  to obtain those representing  $x_1+x_2$  in the  $\heartsuit$ -scheme; apply the same routine for the remaining elements to compute the final sum.

Notice that we described the protocol for a result in  $\mathbb{Z}/3\mathbb{Z}$  but it is easily adaptable for a result in, let say,  $\mathbb{Z}/q\mathbb{Z}$ . Indeed, during the first step, we add a single  $\clubsuit$  to the first commitment and a single  $\heartsuit$  to the second; thus for a sum that could be equal to  $q - 1$ , we add  $q - 2$   $\clubsuit$ s to the first commitment and  $q - 2$   $\heartsuit$ s to the second.

### 3 ZKP Protocol for Usowan

We present a card-based ZKP protocol for Usowan. Consider an Usowan instance composed as a rectangular grid of size  $p \times q$ .

#### 3.1 Setup Phase

The verifier  $V$  and prover  $P$  place black commitments on each cell of the  $p \times q$  grid (also on the numbered cells) and place red commitments (“dummy” commitments) on the left of the first column and below the last row so that we have  $(p + 1)(q + 1)$  commitments.

### 3.2 Connectivity Phase

We apply the sub-protocol introduced in Sect. 2.4 to form a white connected figure. After this phase,  $V$  is convinced that the white commitments are connected (rule 2). Moreover,  $V$  reveals the commitments corresponding to numbered cells to check that they are indeed white (rule 1). Notice that revealing directly those commitments does reveal information about the solution (*i.e.*,  $V$  learns that those cells are white), but this information is already known independently of the protocol.

### 3.3 Verification Phases

There are two rules to check: black commitments cannot touch horizontally nor vertically (rule 3) and each numbered cell has the corresponding number of black cells around it except for one *liar* in each region (rule 4).

*Lonely Black.* For each pair of adjacent commitments,  $V$  applies the five-card trick introduced in Sect. 2.3 to the two commitments to compute their disjunction. We consider here that a white commitment is equal to 1 while a black commitment is equal to 0 (see the encoding (2)). Hence, if the output is 1 then it means that at least one commitment is white so  $V$  continues, otherwise  $V$  aborts (because the only case of output 0 is when there are two black commitments).

*Liar.*  $V$  needs to check that each numbered cell has the corresponding number of black cells around it except for exactly one *liar* in each region. We cannot simply check the number of black cells because it leaks information. Instead, we compute the sum of black cells in  $\mathbb{Z}/5\mathbb{Z}$  introduced in Sect. 2.5 for all numbered cells in a region. However, we do not directly reveal the result but just the  $(x-1)$ -st card of the output sequence. This ensures that the sum is equal or not to  $x$  instead of giving the actual sum.

It remains one sub-protocol to use because the addition is destructive; thus, we need to copy commitments sharing a numbered cell. The copy protocol is described in Sect. 2.2. We can now formally describe the liar verification. For every region, apply the following steps:

1. For each cell that shares  $k > 1$  numbered cells, apply the copy protocol (introduced in Sect. 2.2)  $k - 1$  times.
2. For each numbered cell, compute the addition of its four neighbors<sup>3</sup>. Recall that the result is encoded as the  $\heartsuit$ -scheme (see Sect. 2); thus, the result of the sum has a  $\heartsuit$  in its corresponding position (and all other cards are  $\clubsuit$ s).
3. For each sequence obtained in the previous step, pick the card in the position that corresponds to the number written on the numbered cell. The result must be kept secret (*i.e.*, keep the cards face-down).

---

<sup>3</sup> For a numbered cell in the edge of the board, compute the addition of its three or two neighbors.

Example:

$$\begin{array}{|c|} \hline b \\ \hline \end{array}
 \begin{array}{|c|} \hline a \\ \hline \end{array}
 \begin{array}{|c|} \hline 3 \\ \hline \end{array}
 \begin{array}{|c|} \hline c \\ \hline \end{array}
 \longrightarrow a + b + c + d = \begin{array}{|c|c|c|c|c|} \hline ? & ? & ? & ? & ? \\ \hline \end{array}$$

$\begin{array}{cccccc} & & 0 & 1 & 2 & 3 & 4 \\ & & & & & \uparrow & \end{array}$

4. Shuffle and reveal all the cards previously chosen. If exactly one club is revealed, then continue (*i.e.*, there is exactly one liar); otherwise aborts.

## 4 Conclusion

We propose a physical ZKP protocol for Usowan, which has an interesting rule: some information on the initial grid are incorrect. For verifying such constraints without revealing knowledge about the solution, we construct a protocol based on computing the sum [34]. With this trick we are able to prove that we can hide exactly one liar in each room. The next step will be to see how we can propose a cryptographic ZKP protocol to prove that someone is lying. This is clearly not easy and might require complex and modern cryptographic primitives while we are able to do it only with cards and envelopes in real life.

**Acknowledgements.** We thank the anonymous referees, whose comments have helped us to improve the presentation of the paper. This work was supported in part by JSPS KAKENHI Grant Numbers JP21K11881 and JP18H05289. This study was partially supported by the French ANR project ANR-18-CE39-0019 (MobiS5). Other programs also fund to write this paper, namely the French government research program “Investissements d’Avenir” through the IDEX-ISITE initiative 16-IDEX-0001 (CAP 20-25) and the IMobS3 Laboratory of Excellence (ANR-10-LABX-16-01). Finally, the French ANR project DECRYPT (ANR-18-CE39-0007) and SEVERITAS (ANR-20-CE39-0009) also subsidize this work.

## A Mizuki–Sone Copy Protocol [22]

The protocol proceeds as follows.<sup>4</sup>

1. Turn over all face-up cards and put the commitment to  $a$  above the four additional cards as follows:

$$\begin{array}{|c|c|} \hline ? & ? \\ \hline \end{array}
 \begin{array}{|c|} \hline \clubsuit \\ \hline \end{array}
 \begin{array}{|c|} \hline \heartsuit \\ \hline \end{array}
 \begin{array}{|c|} \hline \heartsuit \\ \hline \end{array}
 \begin{array}{|c|} \hline \clubsuit \\ \hline \end{array}
 \longrightarrow \begin{array}{|c|c|c|c|} \hline ? & ? & ? & ? \\ \hline \end{array}$$

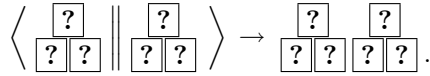
$\begin{array}{cccc} & & a & \\ & & \downarrow & \\ & & \underbrace{\hspace{2cm}} & \\ & & 0 & 1 \end{array}$

Note that black-to-red represents 0, and red-to-black represents 1 according to Eq. (2).

---

<sup>4</sup> This description is a compact version of the original one [22]. Here, we use a pile-shifting shuffle in step 2 instead of using a random bisection cut invented in [22].

2. Apply a pile-shifting shuffle as follows:



3. Reveal the two above cards and obtain two commitments to  $a$  as follows (note that negating a commitment is easy).

- (a) If they are  $\heartsuit \heartsuit$ , then the four bottom cards are  $\underbrace{??}_{a} \underbrace{??}_{\bar{a}}$ .
- (b) If they are  $\heartsuit \clubsuit$ , then the four bottom cards are  $\underbrace{??}_{\bar{a}} \underbrace{??}_{a}$ .

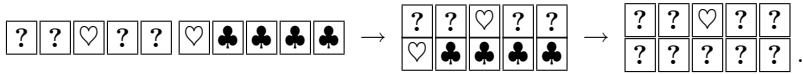
### B Input-preserving Five-Card Trick [16]

The sub-protocol proceeds as follows.

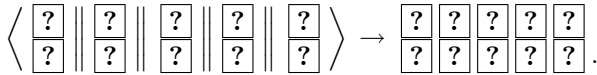
1. Add helping cards and swap the two cards of the commitment to  $a$  so that we have the negation  $\bar{b}$ , as follows:



2. Rearrange the sequence of cards and turn over the face-up cards as:



3. Regarding cards in the same column as a pile, apply a pile-shifting shuffle to the sequence:



- 4. Reveal all the cards in the above row.
  - (a) If the resulting sequence is  $\clubsuit \clubsuit \heartsuit \heartsuit \heartsuit \heartsuit$  (up to cyclic shifts), then  $a \vee b = 0$ .
  - (b) If it is  $\heartsuit \clubsuit \heartsuit \clubsuit \heartsuit$  (up to cyclic shifts), then  $a \vee b = 1$ .
- 5. After turning over all the face-up cards, apply a pile-shifting shuffle.
- 6. Reveal all the cards in the bottom row; then, the revealed cards should include exactly one  $\heartsuit$ .
- 7. Shift the sequence of piles so that the leftmost card is the revealed  $\heartsuit$  and swap the two cards of the commitment to  $\bar{b}$  to restore commitments to  $a$  and  $b$ .

## C How to Form a White Polyomino

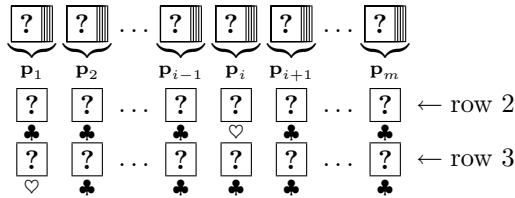
Before explaining the protocol, we need to describe two crucial sub-protocols first, namely the chosen pile protocol and the 4-neighbour protocol.

### C.1 Chosen Pile Protocol [9]

This protocol allows  $P$  to choose a pile of cards without  $V$  knowing which one it is. Some operations can be done on this pile while all the commitments are replaced in their initial order.

This protocol is an extended version of the “chosen pile cut” proposed in [14]. Given  $m$  piles  $(\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_m)$  with  $2m$  additional cards, the *chosen pile protocol* enables a prover  $P$  to choose the  $i$ -th pile  $\mathbf{p}_i$  (without revealing the index  $i$ ) and revert the sequence of  $m$  piles to their original order after applying other operations to  $p_i$ .

- Using  $m - 1$   $\clubsuit$ s and one  $\heartsuit$ ,  $P$  places  $m$  face-down cards (denoted by *row 2*) below the given piles such that only the  $i$ -th card is  $\heartsuit$ . We further put  $m$  cards (denoted by *row 3*) below the cards such that only the first card is  $\heartsuit$ :



- Considering the cards in the same column as a pile, apply a pile-shifting shuffle to the sequence of piles.
- Reveal all the cards in *row 2*. Then, exactly one  $\heartsuit$  appears, and the pile above the revealed  $\heartsuit$  is the  $i$ -th pile (thus  $P$  can obtain  $\mathbf{p}_i$ ). After this step is invoked, other operations are applied to the chosen pile. Then, the chosen pile is placed back to the  $i$ -th position in the sequence.
- Remove the revealed cards, *i.e.*, the cards in *row 2*. (Note, therefore, that we do not use the card  $\heartsuit$  revealed in Step 3.) Then, apply a pile-shifting shuffle.
- Reveal all the cards in *row 3*. Then, one  $\heartsuit$  appears, and the pile above the revealed  $\heartsuit$  is  $\mathbf{p}_1$ . Therefore, by shifting the sequence of piles (such that  $\mathbf{p}_1$  becomes the leftmost pile in the sequence), we can obtain a sequence of piles whose order is the same as the original one without revealing any information about the order of the input sequence.

### C.2 Sub-protocol: 4-Neighbour Protocol [27]

Given  $pq$  commitments placed on a  $p \times q$  grid, a prover  $P$  has a commitment in mind, which we call a *target* commitment. The prover  $P$  wants to reveal the target commitment and another one that lies next to the target commitment

(without revealing their exact positions). Here, a verifier  $V$  should be convinced that the second commitment is a neighbour of the first one (without knowing which one) as well as  $V$  should be able to confirm the colours of both the commitments. To handle the case where the target commitment is at the edge of the grid, we place commitments to red (as “dummy” commitments) in the left of the first column and the below of the last row to prevent  $P$  from choosing a commitment that is not a neighbour. Thus, the size of the expanded grid is  $(p + 1) \times (q + 1)$ .<sup>5</sup>

This sub-protocol proceeds as follows.

1.  $P$  and  $V$  pick the  $(p + 1)(q + 1)$  commitments on the grid from left-to-right and top-to-bottom to make a sequence of commitments:

$$\boxed{?} \boxed{?} \quad \boxed{?} \boxed{?} \quad \boxed{?} \boxed{?} \quad \boxed{?} \boxed{?} \quad \cdots \quad \boxed{?} \boxed{?}.$$

2.  $P$  uses the chosen pile protocol (Sect. 2) to reveal the target commitment.
3.  $P$  and  $V$  pick all the four neighbours of the target commitment. Since a pile-shifting shuffle is a cyclic reordering, the distance between commitments are kept (up to a given modulo). That is, for a target commitment (not at the edge), the possible four neighbours are at distance one for the left or right one, and  $p + 1$  for the bottom or top one. Therefore,  $P$  and  $V$  can determine the positions of all the four neighbours.
4. Among these four neighbours,  $P$  chooses one commitment using the chosen pile protocol and reveals it.
5.  $P$  and  $V$  end the second and first chosen pile protocols.

### C.3 Full Protocol

Assume that there is a grid having  $p \times q$  cells. Without loss of generality,  $P$  wants to arrange white commitments on the grid such that they form a white-polyomino while  $V$  is convinced that the placement of commitments is surely a white-polyomino. The method is as follows.

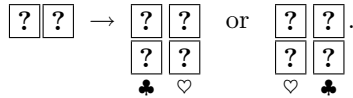
1.  $P$  and  $V$  place a commitment to black (*i.e.*,  $\clubsuit \heartsuit$ ) on every cell and commitments to red as mentioned in Sect. 2.4 so that they have  $(p + 1)(q + 1)$  commitments on the board.
2.  $P$  uses the chosen pile protocol to choose one black commitment that  $P$  wants to change.
  - (a)  $V$  swaps the two cards constituting the chosen commitment so that it becomes a white commitment (recall the encoding (1)).

---

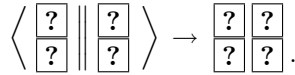
<sup>5</sup> Here, we do not place dummy commitments in the row above the first one and in the column right to the last one because in the expanded grid of size  $(p + 1)(q + 1)$  the row above the first one can be regarded as the last row, *i.e.*, dummy commitments. Thus, we do not need dummy commitments placed in the row above the first one, which also holds for the column right to the last one.



- (b)  $P$  and  $V$  end the chosen pile protocol to return the commitments to their original positions.
- 3.  $P$  and  $V$  repeat the following steps exactly  $pq - 1$  times.
  - (a)  $P$  chooses one white commitment as a target and one black commitment among its neighbours using the 4-neighbour protocol; the neighbour is chosen such that  $P$  wants to make it white.
  - (b)  $V$  reveals the target commitment. If it corresponds to white, then  $V$  continues; otherwise  $V$  aborts.
  - (c)  $V$  reveals the neighbour commitment (chosen by  $P$ ). If it corresponds to black, then  $P$  makes the neighbour white or keep it black (depending on  $P$ 's choice) by executing the following steps; otherwise  $V$  aborts.
    - i. If  $P$  wants to change the commitment,  $P$  places face-down club-to-heart pair below it; otherwise,  $P$  places a heart-to-club pair:



- ii. Regarding cards in the same column as a pile,  $V$  applies a pile-shifting shuffle to the sequence of piles:



- iii.  $V$  reveals the two cards in the second row. If the revealed right card is  $\heartsuit$ , then  $V$  swaps the two cards in the first row; otherwise  $V$  does nothing.
- (d)  $P$  and  $V$  end the 4-neighbour protocol.
- 4.  $P$  and  $V$  remove all the red commitments (*i.e.*, dummy commitments) so that we have  $pq$  commitments on the board.

After this process,  $V$  is convinced that all the white commitments represent a white-polyomino. Therefore, this method allows a prover  $P$  to make a solution that only  $P$  has in mind, guaranteed to satisfy the connectivity constraint.

If the number of white cells in the final polyomino, say  $k$ , is public to a verifier  $V$ , it is sufficient that in Step 3,  $P$  and  $V$  repeat  $k - 1$  times and in Step 3c, and hence,  $V$  simply swaps the two cards constituting the neighbour commitment to make it white (without  $P$ 's choice).

## D Security Proofs



Our protocol needs to verify three security properties given as theorems. Note that the sub-protocols used from the literature have been proven secure *i.e.*, they are correct, complete, sound and zero-knowledge.

**Theorem 1. (Completeness).** *If  $P$  knows the solution of an Usowan grid, then  $P$  can convince  $V$ .*



*Proof.*  $P$  convinces  $V$  in the sense that the protocol does not abort which means that all the rules are satisfied. The protocol can be split into two phases: (1) the connectivity phase and (2) the verification phase.

(1) Since  $P$  knows the solution, the white cells are connected and hence  $P$  can always choose a black commitment at step 2 to swap it to white.

(2) For the lonely black verification, there is no configuration of two black cells that are touching horizontally nor vertically hence for every pair of adjacent cells, there is always at least one white cell.








For the liar verification, there is exactly (in each region) one numbered cell surrounded by a different number of black cells. Suppose, without loss of generality, that the liar cell is equal to  $i$  in a given region (the same result could be applied for each other region). When the sum of the four neighbours is done, the card at position (from left)  $i + 1$  is  otherwise the numbered card is not a liar. Thus when revealing the cards at the last step, there is always a  card.

**Theorem 2. (Soundness).** *If  $P$  does not provide a solution of the  $p \times q$  Usowan grid,  $P$  is not able to convince  $V$ .*

*Proof.* Suppose that  $P$  does not provide a solution. If the white cells are not connected, then  $P$  cannot choose a neighbor commitment that  $P$  wants to change at step 3c. If there are two black commitments touching (or more), then the five-card trick will output 0; hence,  $V$  will abort. Finally, if there is not one liar exactly in a given region, then the last step of the verification will reveal either no  or at least two s; hence,  $V$  will abort.

**Theorem 3. (Zero-knowledge).**  *$V$  learns nothing about  $P$ 's solution of the given grid  $G$ .*

*Proof.* We use the same proof technique as in [11], namely the description of an efficient *simulator* that simulates the interaction between an honest prover and a cheating verifier. The goal is to produce an indistinguishable interaction from the verifier's view (with the prover). Notice that the simulator does not have the solution but it can swap cards during shuffles. Informally, the verifier cannot distinguish between the distributions of two protocols, one that is run with the actual solution and one with random commitments. The simulator acts as follows.

- The simulator constructs a random connected white polyomino.
- During the lonely black verification, the simulator replaces the cards in the five-card trick introduced in Sect. 2.3 with . While the latter sequence is randomly shifted, this ensures that the protocol continues.
- During the liar verification, the simulator simply replaces, in the last step, the cards to have exactly one  and the rest as s. This ensures that there is exactly one liar in a given region, meaning that the protocol does not abort.

The simulated and real proofs are indistinguishable and hence  $V$  learns nothing from the connectivity and verification phases. Finally, we conclude that the protocol is zero-knowledge.

## References

1. <https://www.nikoli.co.jp/en/puzzles/usowan.html> . Nikoli, Usowan
2. Abe, Y., Iwamoto, M., Ohta, K.: Efficient private PEZ protocols for symmetric functions. In: Hofheinz, D., Rosen, A. (eds.) TCC 2019. LNCS, vol. 11891, pp. 372–392. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-36030-6\\_15](https://doi.org/10.1007/978-3-030-36030-6_15)
3. Balogh, J., Csirik, J.A., Ishai, Y., Kushilevitz, E.: Private computation using a PEZ dispenser. *Theor. Comput. Sci.* **306**(1–3), 69–84 (2003). [https://doi.org/10.1016/S0304-3975\(03\)00210-X](https://doi.org/10.1016/S0304-3975(03)00210-X)
4. 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](https://doi.org/10.1007/3-540-46885-4_23)
5. Bultel, X., Dreier, J., Dumas, J., Lafourcade, P.: Physical zero-knowledge proofs for Akari, Takuzu, Kakuro and KenKen. In: Demaine, E.D., Grandoni, F. (eds.) Fun with Algorithms. LIPIcs, vol. 49, pp. 8:1–8:20. Schloss Dagstuhl, Dagstuhl (2016). <https://doi.org/10.4230/LIPIcs.FUN.2016.8>
6. Bultel, X., et al.: Physical zero-knowledge proof for Makaro. In: Izumi, T., Kuznetsov, P. (eds.) SSS 2018. LNCS, vol. 11201, pp. 111–125. Springer, Cham (2018). [https://doi.org/10.1007/978-3-030-03232-6\\_8](https://doi.org/10.1007/978-3-030-03232-6_8)
7. Chien, Y.-F., Hon, W.-K.: Cryptographic and physical zero-knowledge proof: from Sudoku to Nonogram. In: Boldi, P., Gargano, L. (eds.) FUN 2010. LNCS, vol. 6099, pp. 102–112. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-13122-6\\_12](https://doi.org/10.1007/978-3-642-13122-6_12)
8. Dreier, J., Jonker, H., Lafourcade, P.: Secure auctions without cryptography. In: Ferro, A., Luccio, F., Widmayer, P. (eds.) Fun with Algorithms. LNCS, vol. 8496, pp. 158–170. Springer, Cham (2014). [https://doi.org/10.1007/978-3-319-07890-8\\_14](https://doi.org/10.1007/978-3-319-07890-8_14)
9. Dumas, J.G., Lafourcade, P., Miyahara, D., Mizuki, T., Sasaki, T., Sone, H.: Interactive physical zero-knowledge proof for Norinori. In: Du, D.Z., Duan, Z., Tian, C. (eds.) Computing and Combinatorics. LNCS, vol. 11653, pp. 166–177. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-26176-4\\_14](https://doi.org/10.1007/978-3-030-26176-4_14)
10. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems. In: STOC 1985, pp. 291–304. ACM, New York (1985). <https://doi.org/10.1145/22145.22178>
11. Gradwohl, R., Naor, M., Pinkas, B., Rothblum, G.N.: Cryptographic and physical zero-knowledge proof systems for solutions of Sudoku puzzles. *Theory Comput. Syst.* **44**(2), 245–268 (2009). <https://doi.org/10.1007/s00224-008-9119-9>
12. Isuzugawa, R., Miyahara, D., Mizuki, T.: Zero-knowledge proof protocol for Cryptarithmic using dihedral cards. In: Kostitsyna, I., Orponen, P. (eds.) UCNC 2021. LNCS, vol. 12984, pp. 51–67. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-87993-8\\_4](https://doi.org/10.1007/978-3-030-87993-8_4)
13. Iwamoto, C., Haruishi, M.: Computational complexity of Usowan puzzles. *IEICE Trans. Fundamentals E101.A*, 1537–1540 (2018). <https://doi.org/10.1587/transfun.E101.A.1537>
14. Koch, A., Walzer, S.: Foundations for actively secure card-based cryptography. In: Farach-Colton, M., Prencipe, G., Uehara, R. (eds.) Fun with Algorithms. LIPIcs, vol. 157, pp. 17:1–17:23. Schloss Dagstuhl, Dagstuhl (2021). <https://doi.org/10.4230/LIPIcs.FUN.2021.17>
15. Lafourcade, P., Miyahara, D., Mizuki, T., Robert, L., Sasaki, T., Sone, H.: How to construct physical zero-knowledge proofs for puzzles with a “single loop” condition. *Theor. Comput. Sci.* **888**, 41–55 (2021). <https://doi.org/10.1016/j.tcs.2021.07.019>

16. Miyahara, D., et al.: Card-based ZKP protocols for Takuzu and Juosan. In: Farach-Colton, M., Prencipe, G., Uehara, R. (eds.) *Fun with Algorithms. LIPIcs*, vol. 157, pp. 20:1–20:21. Schloss Dagstuhl, Dagstuhl (2021). <https://doi.org/10.4230/LIPIcs.FUN.2021.20>
17. Miyahara, D., Sasaki, T., Mizuki, T., Sone, H.: Card-based physical zero-knowledge proof for Kakuro. *IEICE Trans. Fundamentals* **102-A**(9), 1072–1078 (2019). <https://doi.org/10.1587/transfun.E102.A.1072>
18. 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](https://doi.org/10.1007/978-3-319-48965-0_29)
19. Mizuki, T., Kugimoto, Y., Sone, H.: Secure multiparty computations using a dial lock. In: Cai, J.-Y., Cooper, S.B., Zhu, H. (eds.) *TAMC 2007. LNCS*, vol. 4484, pp. 499–510. Springer, Heidelberg (2007). [https://doi.org/10.1007/978-3-540-72504-6\\_45](https://doi.org/10.1007/978-3-540-72504-6_45)
20. Mizuki, T., Kugimoto, Y., Sone, H.: Secure multiparty computations using the 15 puzzle. In: Dress, A., Xu, Y., Zhu, B. (eds.) *COCOA 2007. LNCS*, vol. 4616, pp. 255–266. Springer, Heidelberg (2007). [https://doi.org/10.1007/978-3-540-73556-4\\_28](https://doi.org/10.1007/978-3-540-73556-4_28)
21. Mizuki, T., Shizuya, H.: Practical card-based cryptography. In: *Fun with Algorithms. LNCS*, vol. 8496, pp. 313–324. Springer, Cham (2014). [https://doi.org/10.1007/978-3-319-07890-8\\_27](https://doi.org/10.1007/978-3-319-07890-8_27)
22. 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](https://doi.org/10.1007/978-3-642-02270-8_36)
23. Moran, T., Naor, M.: Polling with physical envelopes: a rigorous analysis of a human-centric protocol. In: Vaudenay, S. (ed.) *EUROCRYPT 2006. LNCS*, vol. 4004, pp. 88–108. Springer, Heidelberg (2006). [https://doi.org/10.1007/11761679\\_7](https://doi.org/10.1007/11761679_7)
24. Moran, T., Naor, M.: Split-ballot voting: everlasting privacy with distributed trust. In: Ning, P., di Vimercati, S.D.C., Syverson, P.F. (eds.) *ACM Conference on Computer and Communications Security*, pp. 246–255. ACM (2007). <https://doi.org/10.1145/1315245.1315277>
25. Moran, T., Naor, M.: Basing cryptographic protocols on tamper-evident seals. *Theor. Comput. Sci.* **411**(10), 1283–1310 (2010). <https://doi.org/10.1016/j.tcs.2009.10.023>
26. Nishimura, A., Hayashi, Y., Mizuki, T., Sone, H.: Pile-shifting scramble for card-based protocols. *IEICE Trans. Fundamentals* **101-A**(9), 1494–1502 (2018). <https://doi.org/10.1587/transfun.E101.A.1494>
27. Robert, L., Miyahara, D., Lafourcade, P., Mizuki, T.: Card-based ZKP for connectivity: applications to Nurikabe, Hitori, and Heyawake. *New Gener. Comput.* **40**, 149–171 (2022). <https://doi.org/10.1007/s00354-022-00155-5>
28. Robert, L., Miyahara, D., Lafourcade, P., Libralesso, L., Mizuki, T.: Physical zero-knowledge proof and NP-completeness proof of Suguru puzzle. *Inf. Comput.* **285**, 1–14 (2022). <https://doi.org/10.1016/j.ic.2021.104858>
29. Ruangwises, S.: An improved physical ZKP for nonogram. In: Du, D.-Z., Du, D., Wu, C., Xu, D. (eds.) *COCOA 2021. LNCS*, vol. 13135, pp. 262–272. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-92681-6\\_22](https://doi.org/10.1007/978-3-030-92681-6_22)
30. Ruangwises, S.: Two standard decks of playing cards are sufficient for a ZKP for sudoku. *New Gen. Comput.* 1–17 (2021). <https://doi.org/10.1007/s00354-021-00146-y>

31. Ruangwises, S., Itoh, T.: Physical zero-knowledge proof for numberlink puzzle and  $k$  vertex-disjoint paths problem. *New Gen. Comput.* **39**(1), 3–17 (2020). <https://doi.org/10.1007/s00354-020-00114-y>
32. Ruangwises, S., Itoh, T.: Physical zero-knowledge proof for Ripple Effect. *Theor. Comput. Sci.* **895**, 115–123 (2021). <https://doi.org/10.1016/j.tcs.2021.09.034>
33. Ruangwises, S., Itoh, T.: Physical ZKP for connected spanning subgraph: applications to bridges puzzle and other problems. In: Kostitsyna, I., Orponen, P. (eds.) *UCNC 2021*. LNCS, vol. 12984, pp. 149–163. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-87993-8\\_10](https://doi.org/10.1007/978-3-030-87993-8_10)
34. Ruangwises, S., Itoh, T.: Securely computing the  $n$ -variable equality function with  $2n$  cards. *Theor. Comput. Sci.* **887**, 99–110 (2021). <https://doi.org/10.1016/j.tcs.2021.07.007>
35. Ruangwises, S., Itoh, T.: Physical ZKP for Makaro using a standard deck of cards. In: Du, D., et al. (eds.) *TAMC 2022*, LNCS 13571. Springer, Cham (2022)
36. Sasaki, T., Miyahara, D., Mizuki, T., Sone, H.: Efficient card-based zero-knowledge proof for Sudoku. *Theor. Comput. Sci.* **839**, 135–142 (2020). <https://doi.org/10.1016/j.tcs.2020.05.036>
37. Shinagawa, K., et al.: Secure computation protocols using polarizing cards. *IEICE Trans. Fundamentals* **E99.A**(6), 1122–1131 (2016). <https://doi.org/10.1587/transfun.E99.A.1122>
38. Shinagawa, K., et al.: Card-based protocols using regular polygon cards. *IEICE Trans. Fundamentals* **100-A**(9), 1900–1909 (2017). <https://doi.org/10.1587/transfun.E100.A.1900>



# Complexity Analysis of a Stochastic Variant of Generalized Alternating Direction Method of Multipliers

Jia Hu<sup>1,2</sup>, Tiande Guo<sup>1</sup>, and Congying Han<sup>1(✉)</sup>

<sup>1</sup> School of Mathematical Sciences, University of Chinese Academy of Sciences, No.19A Yuquan Road, Beijing 100049, People's Republic of China

hujia17@mails.ucas.ac.cn, {tdguo,hancy}@ucas.ac.cn

<sup>2</sup> Networked Supporting Software International S&T Cooperation Base of China, Jiangxi Normal University, Nanchang 330022, People's Republic of China

**Abstract.** Alternating direction method of multipliers (ADMM) receives much attention in the field of optimization and computer science, etc. The generalized ADMM (G-ADMM) proposed by Eckstein and Bertsekas incorporates an acceleration factor and is more efficient than the original ADMM. However, G-ADMM is not applicable in some models where the objective function value (or its gradient) is computationally costly or even impossible to compute. In this paper, we consider the two-block separable convex optimization problem with linear constraints, where only noisy estimations of the gradient of the objective function are accessible. Under this setting, we propose a stochastic linearized generalized ADMM (called SLG-ADMM) where two subproblems are approximated by some linearization strategies. By properly choosing algorithm parameters, we show, for objective function value gap and constraint violation, the worst-case  $\mathcal{O}(1/\sqrt{k})$  and  $\mathcal{O}(\ln k/k)$  convergence rates in expectation measured by the iteration complexity for general convex and strongly convex problems respectively ( $k$  represents the iteration counter). For the latter case, we also obtain the convergence of the ergodic iterates generated by the proposed SLG-ADMM.

**Keywords:** Alternating direction method of multipliers (ADMM) · Iteration complexity · Stochastic approximation

## 1 Introduction

We consider the following two-block separable convex optimization problem with linear equality constraints:

$$\min \{ \theta_1(x) + \theta_2(y) \mid Ax + By = b, x \in \mathcal{X}, y \in \mathcal{Y} \}, \quad (1)$$

---

This work was supported by Science and Technology Project of SGCC (5700-202055486A-0-0-00).

where  $A \in \mathbb{R}^{n \times n_1}$ ,  $B \in \mathbb{R}^{n \times n_2}$ ,  $b \in \mathbb{R}^n$ ,  $\mathcal{X} \subseteq \mathbb{R}^{n_1}$ , and  $\mathcal{Y} \subseteq \mathbb{R}^{n_2}$  are closed convex sets, and  $\theta_2 : \mathbb{R}^{n_2} \rightarrow \mathbb{R} \cup \{+\infty\}$  is a convex function (not necessarily smooth).  $\theta_1 : \mathbb{R}^{n_1} \rightarrow \mathbb{R}$  is a convex function and is smooth on an open set containing  $\mathcal{X}$ , but has its specific structure; in particular, we assume that there is a stochastic first-order oracle ( $\mathcal{SFO}$ ) for  $\theta_1$ , which returns a stochastic gradient  $G(x, \xi)$  at  $x$ , where  $\xi$  is a random variable whose distribution is supported on  $\Xi \subseteq \mathbb{R}^d$ , satisfying

- a)  $\mathbb{E}[G(x, \xi)] = \nabla\theta_1(x)$ , and
- b)  $\mathbb{E}[\|G(x, \xi) - \nabla\theta_1(x)\|^2] \leq \sigma^2$ ,

where  $\sigma > 0$  is some constant. In addition, we make the following assumptions throughout the paper: (i) The solution set of (1) is assumed to be nonempty. (ii) the gradient of  $\theta_1$  is  $L$ -Lipschitz continuous for some  $L > 0$ , i.e.,  $\|\nabla\theta_1(x) - \nabla\theta_1(y)\| \leq L\|x - y\|$  for any  $x, y \in \mathcal{X}$ . (iii)  $y$ -subproblem has a minimizer at each iteration. As a linearly constrained convex optimization problem, though the model (1) is special, it is rich enough to characterize many optimization problems arising from various application fields, such as machine learning, image processing, and signal processing. In these fields, a typical scenario is where one of the functions represents some data fidelity term, and the other is a regularization term.

Without considering the specific structure, a classical method for solving problem (1) is the alternating direction method of multipliers (ADMM). ADMM was originally proposed by Glowinski and Marrocco [1], and Gabay and Mercier [2], which is a Gauss-Seidel implementation of augmented Lagrangian method [3] or an application of Douglas-Rachford splitting method on the dual problem of (1) [4]. For both convex and non-convex problems, there are extensive studies on the theoretical properties of ADMM. In particular, for convex optimization problems, theoretical results on convergence behavior are abundant, whether global convergence, sublinear convergence rate, or linear convergence rate, see e.g., [4–10]. Recently, ADMM has been studied on nonconvex models satisfying the KL inequality or other similar properties, see e.g., [11–14]. However, when the objective function value (or its gradient) in (1) is computationally costly or even impossible to compute, we can only access some noisy information and deterministic ADMM does not work. Such a setting is exactly what the stochastic programming (SP) model considers. In SP, the objective function is often in the form of expectation. In this case, getting the full function value or gradient information is impractical. To tackle this problem, Robbins and Monro originally introduced the stochastic approximation (SA) approach in 1951 [15]. Since then, SA has gone through many developments; for more detail, readers are referred to a series of works by Nemirovski, Ghadimi, and Lan, etc., see e.g., [16–20]. As for solving problem (1), motivated by the SA, some stochastic ADMM type algorithms have been proposed recently, see e.g., [21–25]. The basic idea in these works is to linearize the function and use stochastic (sub)gradient by SA in the subproblem. In this paper, we will inherit this idea for handling the function  $\theta_1$  in (1) and propose a stochastic linearized general-

ized ADMM, denoted SLG-ADMM, which incorporates an acceleration factor into the subproblem and the update on the dual variables. It should be emphasized that although we treat  $\theta_1$  using similar ideas as in the previous literature, we shall obtain some new results. The proposed SLG-ADMM is based on the well-known generalized ADMM (G-ADMM for short). The G-ADMM was originally proposed by Eckstein and Bertsekas from the perspective of the proximal point algorithm to accelerate the original ADMM. Convergence properties of G-ADMM have been developed in some literature [4, 26], but theoretical analysis of it under the setting of problem (1) is not available and we will bridge the gap between them. To achieve this, we develop the complexity analysis of SLG-ADMM for the general convex and strongly convex problems. For general convex problems, we will establish the worst-case  $\mathcal{O}\left(1/\sqrt{k}\right)$  convergence rate of objective function value gap and constraint violation (not their sum) in expectation for SLG-ADMM with the choices of constant step size and diminishing step size ( $k$  represents the iteration counter, similarly hereinafter). For strongly convex problems, the worst-case convergence rate of SLG-ADMM can be improved to  $\mathcal{O}(\ln k/k)$ . In addition, the convergence of ergodic iterates of SLG-ADMM would be established in the strongly convex case.

The rest of this paper is organized as follows. We present the iterative scheme of SLG-ADMM and summarize some preliminaries which will be used in the theoretical analysis in Sect. 2. In Sect. 3, we derive the worst-case convergence rate for the SLG-ADMM for the general convex and strong convex problems. Finally, we make some conclusions in Sect. 4.

## 2 Stochastic Linearized Generalized ADMM

In this section, we first present the iterative scheme of SLG-ADMM for solving (1), and then we introduce some preliminaries that will be frequently used in the complexity analysis.

We give some remarks on this algorithm. Algorithm 1 is a ADMM type algorithm, which alternates through one  $x$ -subproblem, one  $y$ -subproblem, and an update on the dual variables (multipliers). The algorithm is stochastic since at each iteration  $\mathcal{SFO}$  is called to obtain a stochastic gradient  $G(x^k, \xi)$  which is an unbiased estimation of  $\nabla\theta_1(x^k)$  and is bounded relative to  $\nabla\theta_1(x^k)$  in expectation. The algorithm is linearized because of the following two aspects: (i) The term  $G(x^k, \xi)^T(x - x^k)$  in the  $x$ -subproblem of SLG-ADMM is a stochastic version of linearization of  $\theta_1(x^k)$ . (ii)  $x$ -subproblem and  $y$ -subproblem are added proximal terms  $\frac{1}{2\eta_k}\|x - x^k\|_{G_{1,k}}^2$  and  $\frac{1}{2}\|y - y^k\|_{G_{2,k}}^2$  respectively, where  $\{G_{1,k}\}$  and  $\{G_{2,k}\}$  are two sequences of symmetric and positive definite matrices that can be change with iteration; with the choice of  $G_{2,k} \equiv \tau I_{n_2} - \beta B^T B$ ,  $\tau > \beta\|B^T B\|$ , the quadratic term in the  $y$ -subproblem is linearized. The same fact applies to the  $x$ -subproblem. Furthermore, SLG-ADMM incorporates an acceleration factor  $\alpha$ ; generally, the case with  $\alpha \in (1, 2)$  could lead to better numerical results than the special case with  $\alpha = 1$ . When  $\alpha = 1$ ,  $G_{1,k} \equiv I_{n_1}$ , and the



term  $\frac{1}{2} \|y - y^k\|_{G_{2,k}}^2$  vanishes, SLG-ADMM reduces to the algorithm appeared in earlier literatures [21, 25].

**Algorithm 1: Stochastic Linearized Generalized ADMM (SLG-ADMM)**

Initialize  $x^0 \in \mathcal{X}, y^0 \in \mathcal{Y}, \lambda^0, \alpha \in (0, 2)$ , a positive sequence  $\{\eta_k\}$  and two sequences of symmetric and positive definite matrices:  $\{G_{1,k}\}$  and  $\{G_{2,k}\}$ .  
for  $k = 0, 1, \dots$

    Call the  $\mathcal{SFO}$  to obtain  $G(x^k, \xi)$ .

$$x^{k+1} = \arg \min_{x \in \mathcal{X}} \left\{ G(x^k, \xi)^T (x - x^k) - x^T A^T \lambda^k + \frac{\beta}{2} \|Ax + By^k - b\|^2 + \frac{1}{2\eta_k} \|x - x^k\|_{G_{1,k}}^2 \right\}$$

$$y^{k+1} = \arg \min_{y \in \mathcal{Y}} \left\{ \theta_2(y) - y^T B^T \lambda^k + \frac{\beta}{2} \|\alpha Ax^{k+1} + (1 - \alpha)(b - By^k) + By - b\|^2 + \frac{1}{2} \|y - y^k\|_{G_{2,k}}^2 \right\}$$

$$\lambda^{k+1} = \lambda^k - \beta (\alpha Ax^{k+1} + (1 - \alpha)(b - By^k) + By^{k+1} - b)$$

end

Solving (1) is equivalent to solving the following variational inequality problem: Finding  $w^* = (x^*, y^*, \lambda^*)^1 \in \Omega := \mathcal{X} \times \mathcal{Y} \times \mathbb{R}^n$  such that

$$\theta(w) - \theta(w^*) + (w - w^*)^T F(w^*) \geq 0, \forall w \in \Omega,$$

where

$$u = \begin{pmatrix} x \\ y \end{pmatrix}, w = \begin{pmatrix} x \\ y \\ \lambda \end{pmatrix}, F(w) = \begin{pmatrix} -A^T \lambda \\ -B^T \lambda \\ Ax + By - b \end{pmatrix}, \text{ and } \theta(u) = \theta_1(x) + \theta_2(y).$$

The variables with superscript or subscript such as  $u^k, w^k, \bar{u}_k, \bar{w}_k$  are denoted similarly. In addition, we define two auxiliary sequences for the convergence analysis. More specifically, for the sequence  $\{w^k\}$  generated by the SLG-ADMM, let

$$\tilde{w}^k = \begin{pmatrix} \tilde{x}^k \\ \tilde{y}^k \\ \tilde{\lambda}^k \end{pmatrix} = \begin{pmatrix} x^{k+1} \\ y^{k+1} \\ \lambda^k - \beta (Ax^{k+1} + By^k - b) \end{pmatrix} \text{ and } \tilde{u}^k = \begin{pmatrix} \tilde{x}^k \\ \tilde{y}^k \end{pmatrix}. \quad (2)$$

Based on the above notations and the update scheme of  $\lambda^k$  in SLG-ADMM, we have

$$\lambda^{k+1} - \tilde{\lambda}^k = (1 - \alpha) (\lambda^k - \tilde{\lambda}^k) + \beta B (y^k - \tilde{y}^k) \quad (3)$$

<sup>1</sup> We sometimes use  $(x, y, \lambda)$  to denote  $(x^T, y^T, \lambda^T)^T$ .

and

$$\lambda^k - \lambda^{k+1} = \alpha \left( \lambda^k - \tilde{\lambda}^k \right) + \beta B \left( \tilde{y}^k - y^k \right). \tag{4}$$

Then we get

$$w^k - w^{k+1} = M \left( w^k - \tilde{w}^k \right), \tag{5}$$

where  $M$  is defined as

$$\begin{pmatrix} I_{n_1} & 0 & 0 \\ 0 & I_{n_2} & 0 \\ 0 & -\beta B & \alpha I_n \end{pmatrix}. \tag{6}$$

For notational simplicity, we define two sequences of matrices that will be used later: for  $k = 0, 1, \dots$

$$H_k = \begin{pmatrix} \frac{1}{\eta_k} G_{1,k} & 0 & 0 \\ 0 & \frac{\beta}{\alpha} B^T B + G_{2,k} & \frac{1-\alpha}{\alpha} B^T \\ 0 & \frac{1-\alpha}{\alpha} B & \frac{1}{\beta \alpha} I_n \end{pmatrix}, Q_k = \begin{pmatrix} \frac{1}{\eta_k} G_{1,k} & 0 & 0 \\ 0 & \beta B^T B + G_{2,k} & (1-\alpha) B^T \\ 0 & -B & \frac{1}{\beta} I_n \end{pmatrix}. \tag{7}$$

Obviously, for any  $k$ , the matrices  $M, H_k$ , and  $Q_k$  satisfy  $Q_k = H_k M$ .

At the end of this section, we recall a lemma, which will be used in Sect. 3.

**Lemma 1.** *Suppose function  $f$  is convex and its gradient is  $L$ -Lipschitz continuous, then for any  $x, y, z$  we have*

$$(x - y)^T \nabla f(z) \leq f(x) - f(y) + \frac{L}{2} \|y - z\|^2.$$

In addition, if  $f$  is  $\mu$ -strongly convex, then for any  $x, y, z$  we have

$$(x - y)^T \nabla f(z) \leq f(x) - f(y) + \frac{L}{2} \|y - z\|^2 - \frac{\mu}{2} \|x - z\|^2.$$

*Proof.* See the appendix. □

### 3 Complexity Analysis of SLG-ADMM

In this section, we will establish the worst-case convergence rate of SLG-ADMM. Due to space constraints, proofs are placed in the appendix. For the ease of presentation, we introduce some notations. Denote  $\delta^k = G(x^k, \xi) - \nabla \theta_1(x^k)$ . By the assumptions on  $\mathcal{SF}\mathcal{O}$ , we immediately get  $\mathbb{E}[\delta^k] = 0$  and  $\mathbb{E}[\|\delta^k\|^2] \leq \sigma^2$ . For two matrices  $A$  and  $B$ , the ordering relation  $A \succeq B$  means  $A - B$  is positive semidefinite.  $I_m$  denotes the  $m \times m$  identity matrix. For a vector  $x$ ,  $\|x\|$  denotes its Euclidean norm; for a matrix  $X$ ,  $\|X\|$  denotes its spectral norm.

#### 3.1 A Worst-Case $\mathcal{O}(1/\sqrt{k})$ Convergence Rate

This subsection considers that the function  $\theta_1$  is convex. First, we prove some lemmas. The next lemma gives a key inequality for the sequence  $\{\tilde{w}^k\}$ .

**Lemma 2.** *Let the sequence  $\{w^k\}$  be generated by the SLG-ADMM and the associated  $\{\tilde{w}^k\}$  be defined in (2). Then we have*

$$\theta(u) - \theta(\tilde{u}^k) + (w - \tilde{w}^k)^T F(\tilde{w}^k) \geq (w - \tilde{w}^k)^T Q_k (w^k - \tilde{w}^k) - (x - \tilde{x}^k)^T \delta^k - \frac{L}{2} \|x^k - \tilde{x}^k\|^2, \forall w \in \Omega, \tag{8}$$

where  $Q_k$  is defined in (7).

Next, we need to further explore the terms on the right hand side of (8).

**Lemma 3.** *Let the sequence  $\{w^k\}$  be generated by the SLG-ADMM and the associated  $\{\tilde{w}^k\}$  be defined in (2). Then for any  $w \in \mathcal{X} \times \mathcal{Y} \times \mathbb{R}^n$ , we have*

$$\begin{aligned} & (w - \tilde{w}^k)^T Q_k (w^k - \tilde{w}^k) \\ &= \frac{1}{2} \left( \|w - w^{k+1}\|_{H_k}^2 - \|w - w^k\|_{H_k}^2 \right) + \frac{1}{2\eta_k} \|x^k - \tilde{x}^k\|_{G_{1,k}}^2 + \frac{1}{2} \|y^k - \tilde{y}^k\|_{G_{2,k}}^2 \\ & \quad - \frac{\alpha - 2}{2\beta} \|\lambda^k - \tilde{\lambda}^k\|^2. \end{aligned}$$

Now, we are ready to establish the first main result for SLG-ADMM.

**Theorem 1.** *Let the sequence  $\{w^k\}$  be generated by the SLG-ADMM, the associated  $\{\tilde{w}^k\}$  be defined in (2), and*

$$\bar{w}_k = \frac{1}{k+1} \sum_{t=0}^k \tilde{w}^t.$$

Let  $\{\alpha_k\}$  be any positive sequence and assume that the ordering relation  $\frac{1}{\eta_k} G_{1,k} \succeq \left(\frac{1}{\alpha_k} + L\right) I_{n_1}$  for any  $k$  holds. Then we have

$$\begin{aligned} & \theta(\bar{u}_k) - \theta(u) + (\bar{w}_k - w)^T F(w) \\ & \leq \frac{1}{2(k+1)} \sum_{t=0}^k \left( \|w^t - w\|_{H_t}^2 - \|w^{t+1} - w\|_{H_t}^2 \right) + \frac{1}{k+1} \sum_{t=0}^k (x - x^t)^T \delta^t \\ & \quad + \frac{1}{k+1} \sum_{t=0}^k \frac{\alpha_t}{2} \|\delta^t\|^2. \end{aligned} \tag{9}$$

In SLG-ADMM, appropriate choices of  $G_{1,k}, G_{2,k}$  can make the subproblems easier solved. For  $x$ -subproblem, if we choose  $G_{1,k} = \tau I_{n_1} - \eta_k \beta A^T A, \tau > \eta_k \beta \|A^T A\|$ , then the quadratic term  $\frac{\beta}{2} \|Ax\|^2$  can be eliminated and the resulting solution of this subproblem reduces to a stochastic gradient step. Similarly, with some choice of  $G_{2,k}$ , the  $y$ -subproblem reduces to estimating the resolvent of  $\partial\theta_2$ . In the following corollaries, we choose  $G_{1,k} = \tau I_{n_1} - \eta_k \beta A^T A, \tau > \eta_k \beta \|A^T A\|$  and  $G_{2,k} \equiv G_2$ . Some specific convergence results will be obtained by properly choosing  $\{\eta_t\}$  and  $\{\alpha_t\}$  from Theorem 1.

**Corollary 1.** *Let the sequence  $\{w^k\}$  be generated by the SLG-ADMM, the associated  $\{\tilde{w}^k\}$  be defined in (2), and*

$$\bar{w}_N = \frac{1}{N+1} \sum_{t=0}^N \tilde{w}^t$$

for some pre-selected integer  $N$ . Choosing  $\alpha_k \equiv \frac{1}{\sqrt{N}}$  and  $\eta_k \equiv \frac{1}{\sqrt{N+M}}$ , where  $M$  is a constant satisfying the ordering relation  $\tau(\sqrt{N} + M) I_{n_1} \succeq (\sqrt{N} + L) I_{n_1} + \beta A^T A$ , then we have

$$\begin{aligned} & \mathbb{E} [\|A\bar{x}_N + B\bar{y}_N - b\|] \\ & \leq \frac{1}{2(N+1)} \left( M \|x^0 - x^*\|_{G_{1,0}}^2 + \|y^0 - y^*\|_{\frac{\beta}{\alpha} B^T B + G_2}^2 + \frac{2}{\beta\alpha} (\|\lambda^0 - \lambda^*\|^2 + 1) \right) \\ & \quad + \frac{1}{2\sqrt{N}} (\sigma^2 + \|x^0 - x^*\|_{G_{1,0}}^2) + \frac{1-\alpha}{(N+1)\alpha} (\lambda^0 - \lambda^*)^T B (y^0 - y^*), \end{aligned}$$

and

$$\begin{aligned} & \mathbb{E} [\theta(\bar{u}_N) - \theta(u^*)] \\ & \leq \frac{\|\lambda^*\| + 1}{2(N+1)} \left( M \|x^0 - x^*\|_{G_{1,0}}^2 + \|y^0 - y^*\|_{\frac{\beta}{\alpha} B^T B + G_2}^2 + \frac{2}{\beta\alpha} (\|\lambda^0 - \lambda^*\|^2 + 1) \right) \\ & \quad + \frac{\|\lambda^*\| + 1}{2\sqrt{N}} (\sigma^2 + \|x^0 - x^*\|_{G_{1,0}}^2) + \frac{(1-\alpha)(\|\lambda^*\| + 1)}{(N+1)\alpha} (\lambda^0 - \lambda^*)^T B (y^0 - y^*). \end{aligned}$$

This corollary shows that by choosing a constant step size, the SLG-ADMM achieves both  $\mathbb{E} [\|A\bar{x}_N + B\bar{y}_N - b\|] \leq \varepsilon$  and  $\mathbb{E} [\theta(\bar{u}_N) - \theta(u^*)] \leq \varepsilon$  in  $\mathcal{O}(1/\varepsilon^2)$  number of iterations. Thus, a worst-case  $\mathcal{O}(1/\sqrt{k})$  convergence rate for the SLG-ADMM with the choice of constant size is established. It is worth noting that

$$\mathbb{E} [\theta(\bar{u}_N) - \theta(u^*) + \|A\bar{x}_N + B\bar{y}_N - b\|] \leq \mathcal{O}(1/\sqrt{N})$$

is shown in [21]. In fact, this bound is not strong enough to ensure the expected objective function value gap or constraint violation tend to zero. Afterwards, in [25], the authors establish bounds for  $\mathbb{E} [\|A\bar{x}_N + B\bar{y}_N - b\|]$  and  $\mathbb{E} [\theta(\bar{u}_N) - \theta(u^*)]$  respectively using a lemma in [19], but their result is based on the expectation of ergodic iterates over all random history, which is impractical. Moreover, if  $\theta_1$  is a function of the finite sum form and  $G(x^k, \xi)$  is replaced by  $\nabla\theta_1(x^k)$ , the last two terms in the right hand side of (9) would vanish. This observation indicates a worst-case  $\mathcal{O}(1/k)$  convergence rate of deterministic ADMM.

**Corollary 2.** *Let the sequence  $\{w^k\}$  be generated by the SLG-ADMM, the associated  $\{\tilde{w}^k\}$  be defined in (2), and*

$$\bar{w}_k = \frac{1}{k+1} \sum_{t=0}^k \tilde{w}^t.$$

Choosing  $\alpha_k \equiv \frac{1}{\sqrt{k}}$  and  $\eta_k \equiv \frac{1}{\sqrt{k+M}}$ , where  $M$  is a constant satisfying the ordering relation  $\tau(\sqrt{k} + M) I_{n_1} \succeq (\sqrt{k} + L) I_{n_1} + \beta A^T A$  for any  $k$ , and assuming  $\|w^k - w^*\|_{G_{1,k}}^2 \leq R^2$  for any  $k$ , then we have

$$\begin{aligned} & \mathbb{E} [\|A\bar{x}_k + B\bar{y}_k - b\|] \\ & \leq \frac{1}{2(k+1)} \left( \|y^0 - y^*\|_{\frac{\beta}{\alpha} B^T B + G_2}^2 + \frac{2}{\beta\alpha} (\|\lambda^0 - \lambda^*\|^2 + 1) + MR^2 \right) \\ & \quad + \frac{1}{2\sqrt{k}} (2\sigma^2 + R^2) + \frac{1-\alpha}{(k+1)\alpha} (\lambda^0 - \lambda^*)^T B (y^0 - y^*), \end{aligned}$$

and

$$\begin{aligned} & \mathbb{E} [\theta(\bar{u}_k) - \theta(u^*)] \\ & \leq \frac{\|\lambda^*\| + 1}{2(k+1)} \left( \|y^0 - y^*\|_{\frac{\beta}{\alpha} B^T B + G_2}^2 + \frac{2}{\beta\alpha} (\|\lambda^0 - \lambda^*\|^2 + 1) + MR^2 \right) \\ & \quad + \frac{\|\lambda^*\| + 1}{2\sqrt{k}} (2\sigma^2 + R^2) + \frac{(1-\alpha)(\|\lambda^*\| + 1)}{(k+1)\alpha} (\lambda^0 - \lambda^*)^T B (y^0 - y^*). \end{aligned}$$

This corollary shows that a worst-case  $\mathcal{O}(1/\sqrt{k})$  convergence rate for the SLG-ADMM with the choice of diminishing size is established. One difference is that we require the distance between iterates generated by SLG-ADMM and the solution of (1) to be bounded, a condition that is satisfied in many cases.

### 3.2 A Worst-Case $\mathcal{O}(\ln k/k)$ Convergence Rate Under Strong Convexity

In this section, we assume that  $\theta_1$  is  $\mu$ -strongly convex. With the strong convexity, we can obtain not only the objective function value gap and constraint violation converge to zero in expectation, but also the convergence of ergodic iterates of SLG-ADMM. The main result is presented in the next two theorems. Moreover, we choose  $G_{1,k} = \tau(1 + \eta_k) I_{n_1} - \eta_k \beta A^T A$ ,  $\tau > \beta \|A^T A\|$  and  $G_{2,k} \equiv G_2$ .

**Theorem 2.** *Let the sequence  $\{w^k\}$  be generated by the SLG-ADMM, the associated  $\{\tilde{w}^k\}$  be defined in (2), and*

$$\bar{w}_k = \frac{1}{k+1} \sum_{t=0}^k \tilde{w}^t.$$

Choosing  $\alpha_k \equiv \frac{1}{\mu(k+1)}$  and  $\eta_k \equiv \frac{\tau}{\mu(k+1)+M}$ , where  $M$  is a constant satisfying the ordering relation  $(\tau + M) I_{n_1} \succeq LI_{n_1} + \beta A^T A$  for any  $k$ , and assuming  $\theta_1$

is  $\mu$ -strongly convex, then we have

$$\begin{aligned} & \mathbb{E} [\|A\bar{x}_k + B\bar{y}_k - b\|] \\ \leq & \frac{1}{2(k+1)} \left( \|x^0 - x^*\|_{(\tau+M)I_{n_1} - \beta A^T A}^2 + \|y^0 - y^*\|_{\frac{\beta}{\alpha} B^T B + G_2}^2 \right. \\ & \left. + \frac{2}{\beta\alpha} (\|\lambda^0 - \lambda^*\|^2 + 1) \right) + \frac{\sigma^2}{2\mu(k+1)} (1 + \ln(k+1)) + \frac{1-\alpha}{(k+1)\alpha} (\lambda^0 - \lambda^*)^T B (y^0 - y^*), \end{aligned} \tag{10}$$

and

$$\begin{aligned} & \mathbb{E} [\theta(\bar{u}_k) - \theta(u^*)] \\ \leq & \frac{\|\lambda^*\| + 1}{2(k+1)} \left( \|x^0 - x^*\|_{(\tau+M)I_{n_1} - \beta A^T A}^2 + \|y^0 - y^*\|_{\frac{\beta}{\alpha} B^T B + G_2}^2 + \frac{2}{\beta\alpha} (\|\lambda^0 - \lambda^*\|^2 + 1) \right) \\ & + \frac{\sigma^2 (\|\lambda^*\| + 1)}{2\mu(k+1)} (1 + \ln(k+1)) + \frac{(1-\alpha) (\|\lambda^*\| + 1)}{(k+1)\alpha} (\lambda^0 - \lambda^*)^T B (y^0 - y^*). \end{aligned} \tag{11}$$

This theorem implies that under the assumption that  $\theta_1$  is strongly convex, the worst-case convergence rate for the SLG-ADMM can be improved to  $\mathcal{O}(\ln k/k)$  with the choice of diminishing size.

**Theorem 3.** Let the sequence  $\{w^k\}$  be generated by the SLG-ADMM, the associated  $\{\tilde{w}^k\}$  be defined in (2), and

$$\bar{w}_k = \frac{1}{k+1} \sum_{t=0}^k \tilde{w}^t.$$

Choosing  $\alpha_k \equiv \frac{1}{\mu(k+1)}$  and  $\eta_k \equiv \frac{\tau}{\mu(k+1)+M}$ , where  $M$  is a constant satisfying the ordering relation  $(\tau + M) I_{n_1} \succeq L I_{n_1} + \beta A^T A$  for any  $k$ , and assuming  $\theta_1$  is  $\mu$ -strongly convex,  $B$  is of full column rank, and  $\beta\alpha(k+1) \gg 1$  for some  $k$ , then we have

$$\begin{aligned} & \mathbb{E} \left[ \|\bar{x}_k - x^*\|^2 + \|\bar{y}_k - y^*\|^2 \right] \\ \leq & \left( \frac{2}{\mu} + \frac{4\|A\|^2}{\mu s} \right) (\mathbb{E} [\theta(\bar{u}_k) - \theta(u^*)] + \|\lambda^*\| \mathbb{E} [\|A\bar{x}_k + B\bar{y}_k - b\|]) \\ & + \frac{2}{s} \mathbb{E} [\|A\bar{x}_k + B\bar{y}_k - b\|^2], \end{aligned}$$

where  $s$  denotes the minimum eigenvalue of  $B^T B$ , the bounds for  $\mathbb{E} [\|A\bar{x}_k + B\bar{y}_k - b\|]$  and  $\mathbb{E} [\theta(\bar{u}_k) - \theta(u^*)]$  are the same as in (10) and (11) respectively, and

$$\begin{aligned} & \mathbb{E} [\|A\bar{x}_k + B\bar{y}_k - b\|^2] \\ \leq & \frac{\beta\alpha}{2(\beta\alpha(k+1) - 1)} \left( M \|x^0 - x^*\|_{G_{1,0}}^2 + \|y^0 - y^*\|_{\frac{\beta}{\alpha} B^T B + G_2}^2 + \frac{2}{\beta\alpha} \|\lambda^0 - \lambda^*\|^2 \right) \\ & + \frac{\beta\alpha(k+1)}{2\sqrt{k}(\beta\alpha(k+1) - 1)} \left( \sigma^2 + \|x^0 - x^*\|_{G_{1,0}}^2 \right) + \frac{\beta(1-\alpha)}{\beta\alpha(k+1) - 1} (\lambda^0 - \lambda^*)^T B (y^0 - y^*). \end{aligned}$$

This theorem shows the convergence of ergodic iterates of SLG-ADMM, which is not covered in some earlier literatures [21, 25]. Furthermore, if  $\theta_2$  is also strongly convex, the assumption that  $B$  is of full column rank can be removed.

## 4 Conclusion

In this paper, we propose a stochastic variant of generalized ADMM and establish its worst-case  $\mathcal{O}\left(1/\sqrt{k}\right)$  and  $\mathcal{O}(\ln k/k)$  convergence rates for general convex and strongly convex problems respectively. This result subsumes and improves some existing results established in earlier literature for stochastic ADMM type algorithms. As a by-product, the worst-case convergence rate of the deterministic ADMM algorithm can also be obtained.

## Appendix

### Proof of Lemma 1

*Proof.* Since the gradient of  $f$  is  $L$ -Lipschitz continuous, then for any  $y, z$  we have

$$f(y) \leq f(z) + (y - z)^T \nabla f(z) + \frac{L}{2} \|y - z\|^2.$$

Also, due to the convexity of  $f$ , we have for any  $x, z$

$$f(x) \geq f(z) + (x - z)^T \nabla f(z).$$

Adding the above two inequalities, we get the conclusion. If  $f$  is  $\mu$ -strongly convex, then for any  $x, z$

$$f(x) \geq f(z) + (x - z)^T \nabla f(z) + \frac{\mu}{2} \|x - z\|^2.$$

Then combine this inequality with

$$f(y) \leq f(z) + (y - z)^T \nabla f(z) + \frac{L}{2} \|y - z\|^2,$$

and the proof is completed. □

### Proof of Lemma 2

*Proof.* The optimality condition of the  $x$ -subproblem in SLG-ADMM is

$$\begin{aligned} & (x - x^{k+1})^T \left( G(x^k, \xi) - A^T \lambda^k + \beta A^T (Ax^{k+1} + By^k - b) + \frac{1}{\eta_k} G_{1,k}(x^{k+1} - x^k) \right) \\ & \geq 0, \forall x \in \mathcal{X}. \end{aligned} \tag{12}$$

Using  $\tilde{x}^k$  and  $\tilde{\lambda}^k$  defined in (2) and notation of  $\delta^k$ , (12) can be rewritten as

$$(x - \tilde{x}^k)^T \left( \nabla \theta_1(x^k) + \delta^k - A^T \tilde{\lambda}^k + \frac{1}{\eta_k} G_{1,k}(\tilde{x}^k - x^k) \right) \geq 0, \forall x \in \mathcal{X}. \quad (13)$$

In lemma 1, letting  $y = \tilde{x}^k$ ,  $z = x^k$ , and  $f = \theta_1$ , we get

$$(x - \tilde{x}^k)^T \nabla \theta_1(x^k) \leq \theta_1(x) - \theta_1(\tilde{x}^k) + \frac{L}{2} \|x^k - \tilde{x}^k\|^2. \quad (14)$$

Combining (13) and (14), we obtain

$$\begin{aligned} & \theta_1(x) - \theta_1(\tilde{x}^k) + (x - \tilde{x}^k)^T \left( -A^T \tilde{\lambda}^k \right) \\ & \geq \frac{1}{\eta_k} (x - \tilde{x}^k)^T G_{1,k}(x^k - \tilde{x}^k) - (x - \tilde{x}^k)^T \delta^k - \frac{L}{2} \|x^k - \tilde{x}^k\|^2. \end{aligned} \quad (15)$$

Similarly, the optimality condition of  $y$ -subproblem is

$$\theta_2(y) - \theta_2(\tilde{y}^k) + (y - \tilde{y}^k)^T \left( -B^T \lambda^{k+1} + G_{2,k}(\tilde{y}^k - y^k) \right) \geq 0, \forall y \in \mathcal{Y}. \quad (16)$$

Substituting (3) into (16), we obtain that

$$\begin{aligned} & \theta_2(y) - \theta_2(\tilde{y}^k) + (y - \tilde{y}^k)^T \left( -B^T \tilde{\lambda}^k \right) \\ & \geq (1 - \alpha) (y - \tilde{y}^k)^T B^T (\lambda^k - \tilde{\lambda}^k) + (y - \tilde{y}^k)^T (\beta B^T B + G_{2,k})(y^k - \tilde{y}^k), \forall y \in \mathcal{Y}. \end{aligned} \quad (17)$$

At the same time,

$$\begin{aligned} \tilde{\lambda}^k &= \lambda^k - \beta (A x^{k+1} + B y^{k+1} - b) + \beta B (y^{k+1} - y^k) \\ &= \lambda^k - \beta (A \tilde{x}^k + B \tilde{y}^k - b) + \beta B (\tilde{y}^k - y^k). \end{aligned}$$

That is

$$(\lambda - \tilde{\lambda}^k)^T (A \tilde{x}^k + B \tilde{y}^k - b) = \frac{1}{\beta} (\lambda - \tilde{\lambda}^k)^T (\lambda^k - \tilde{\lambda}^k) + (\lambda - \tilde{\lambda}^k)^T B (\tilde{y}^k - y^k). \quad (18)$$

Combining (15), (17), and (18), we get

$$\begin{aligned} & \theta(u) - \theta(\tilde{u}^k) + \begin{pmatrix} x - \tilde{x}^k \\ y - \tilde{y}^k \\ \lambda - \tilde{\lambda}^k \end{pmatrix}^T \begin{pmatrix} -A^T \tilde{\lambda}^k \\ -B^T \tilde{\lambda}^k \\ A \tilde{x}^k + B \tilde{y}^k - b \end{pmatrix} \\ & \geq \frac{1}{\eta_k} (x - \tilde{x}^k)^T G_{1,k}(x^k - \tilde{x}^k) - (x - \tilde{x}^k)^T \delta^k - \frac{L}{2} \|x^k - \tilde{x}^k\|^2 \\ & \quad + (1 - \alpha) (y - \tilde{y}^k)^T B^T (\lambda^k - \tilde{\lambda}^k) + (y - \tilde{y}^k)^T (\beta B^T B + G_{2,k})(y^k - \tilde{y}^k) \\ & \quad + \frac{1}{\beta} (\lambda - \tilde{\lambda}^k)^T (\lambda^k - \tilde{\lambda}^k) + (\lambda - \tilde{\lambda}^k)^T B (\tilde{y}^k - y^k), \forall w \in \Omega. \end{aligned} \quad (19)$$

Finally, by the definition of  $F$  and  $Q_k$ , we come to the conclusion.  $\square$



**Proof of Lemma 3**

*Proof.* Using  $Q_k = H_k M$  and  $w^k - w^{k+1} = M(w^k - \tilde{w}^k)$  in (5), we have

$$\begin{aligned} (w - \tilde{w}^k)^T Q_k (w^k - \tilde{w}^k) &= (w - \tilde{w}^k)^T H_k M (w^k - \tilde{w}^k) \\ &= (w - \tilde{w}^k)^T H_k (w^k - w^{k+1}). \end{aligned} \quad (20)$$

Now applying the identity: for the vectors  $a, b, c, d$  and a matrix  $H$  with appropriate dimension,

$$(a - b)^T H (c - d) = \frac{1}{2} \left( \|a - d\|_H^2 - \|a - c\|_H^2 \right) + \frac{1}{2} \left( \|c - b\|_H^2 - \|d - b\|_H^2 \right).$$

In this identity, letting  $a = w$ ,  $b = \tilde{w}^k$ ,  $c = w^k$ ,  $d = \tilde{w}^k$ , and  $H = Q_k$ , we have

$$\begin{aligned} (w - \tilde{w}^k)^T H_k (w^k - w^{k+1}) &= \frac{1}{2} \left( \|w - w^{k+1}\|_{H_k}^2 - \|w - w^k\|_{H_k}^2 \right) \\ &\quad + \frac{1}{2} \left( \|w^k - \tilde{w}^k\|_{H_k}^2 - \|w^{k+1} - \tilde{w}^k\|_{H_k}^2 \right). \end{aligned}$$

Next we simplify the term  $\|w^k - \tilde{w}^k\|_{H_k}^2 - \|w^{k+1} - \tilde{w}^k\|_{H_k}^2$ .

$$\begin{aligned} &\|w^k - \tilde{w}^k\|_{H_k}^2 - \|w^{k+1} - \tilde{w}^k\|_{H_k}^2 \\ &= \|w^k - \tilde{w}^k\|_{H_k}^2 - \|w^{k+1} - w^k + w^k - \tilde{w}^k\|_{H_k}^2 \\ &= \|w^k - \tilde{w}^k\|_{H_k}^2 - \|(I_{n_1+n_2+n} - M)(w^k - \tilde{w}^k)\|_{H_k}^2 \\ &= (w^k - \tilde{w}^k)^T \left( H_k - (I_{n_1+n_2+n} - M)^T H_k (I_{n_1+n_2+n} - M) \right) (w^k - \tilde{w}^k) \\ &= (w^k - \tilde{w}^k)^T (H_k M + M^T H_k - M^T H_k M) (w^k - \tilde{w}^k) \\ &= (w^k - \tilde{w}^k)^T ((2I_{n_1+n_2+n} - M^T) Q_k) (w^k - \tilde{w}^k), \end{aligned}$$

where the second equality uses  $w^k - w^{k+1} = M(w^k - \tilde{w}^k)$  in (5), and the last equality holds since the transpose of  $M^T H_k$  is  $H_k M$  and hence

$$\begin{aligned} (w^k - \tilde{w}^k)^T H_k M (w^k - \tilde{w}^k) &= (w^k - \tilde{w}^k)^T M^T H_k (w^k - \tilde{w}^k) \\ &= (w^k - \tilde{w}^k)^T Q_k (w^k - \tilde{w}^k). \end{aligned}$$

The remaining task is to prove

$$\begin{aligned} &(w^k - \tilde{w}^k)^T ((2I_{n_1+n_2+n} - M^T) Q_k) (w^k - \tilde{w}^k) \\ &= \frac{1}{\eta_k} \|x^k - \tilde{x}^k\|_{G_{1,k}}^2 + \|y^k - \tilde{y}^k\|_{G_{2,k}}^2 - \frac{\alpha - 2}{\beta} \|\lambda^k - \tilde{\lambda}^k\|^2. \end{aligned} \quad (21)$$

By simple algebraic operation,

$$(2I_{n_1+n_2+n} - M^T) Q_k = \begin{pmatrix} \frac{1}{\eta_k} G_{1,k} & 0 & 0 \\ 0 & G_{2,k} & (2 - \alpha) B^T \\ 0 & (\alpha - 2) B & \frac{2 - \alpha}{\beta} I_n \end{pmatrix}.$$

With this result, (21) holds and the proof is completed.  $\square$

### Proof of Theorem 1

*Proof.* Combining lemma 2 and lemma 3, we get

$$\begin{aligned}
& \theta(\tilde{u}^t) - \theta(u) + (\tilde{w}^t - w)^T F(\tilde{w}^t) \\
& \leq \frac{1}{2} \left( \|w^t - w\|_{H_t}^2 - \|w^{t+1} - w\|_{H_t}^2 \right) - \frac{1}{2\eta_t} \|x^t - \tilde{x}^t\|_{G_{1,t}}^2 - \frac{1}{2} \|y^t - \tilde{y}^t\|_{G_{2,t}}^2 \\
& \quad + \frac{\alpha - 2}{2\beta} \|\lambda^t - \tilde{\lambda}^t\|^2 + (x - \tilde{x}^t)^T \delta^t + \frac{L}{2} \|x^t - \tilde{x}^t\|^2 \\
& = \frac{1}{2} \left( \|w^t - w\|_{H_t}^2 - \|w^{t+1} - w\|_{H_t}^2 \right) + (x - x^t)^T \delta^t + (x^t - \tilde{x}^t)^T \delta^t \\
& \quad + \frac{1}{2} (x^t - \tilde{x}^t)^T \left( LI_{n_1} - \frac{1}{\eta_t} G_{1,t} \right) (x^t - \tilde{x}^t) - \frac{1}{2} \|y^t - \tilde{y}^t\|_{G_{2,t}}^2 + \frac{\alpha - 2}{2\beta} \|\lambda^t - \tilde{\lambda}^t\|^2 \quad (22) \\
& \leq \frac{1}{2} \left( \|w^t - w\|_{H_t}^2 - \|w^{t+1} - w\|_{H_t}^2 \right) + (x - x^t)^T \delta^t + \frac{\alpha_t}{2} \|\delta^t\|^2 \\
& \quad + \frac{1}{2} (x^t - \tilde{x}^t)^T \left( \left( \frac{1}{\alpha_t} + L \right) I_{n_1} - \frac{1}{\eta_t} G_{1,t} \right) (x^t - \tilde{x}^t) \\
& \leq \frac{1}{2} \left( \|w^t - w\|_{H_t}^2 - \|w^{t+1} - w\|_{H_t}^2 \right) + (x - x^t)^T \delta^t + \frac{\alpha_t}{2} \|\delta^t\|^2,
\end{aligned}$$

where the second inequality holds owing to the Young's inequality and  $\alpha \in (0, 2)$ . Meanwhile,

$$\begin{aligned}
& \frac{1}{k+1} \sum_{t=0}^k \theta(\tilde{u}^t) - \theta(u) + (\tilde{w}^t - w)^T F(\tilde{w}^t) \\
& = \frac{1}{k+1} \sum_{t=0}^k \theta(\tilde{u}^t) - \theta(u) + (\tilde{w}^t - w)^T F(w) \quad (23) \\
& \geq \theta(\bar{u}_k) - \theta(u) + (\bar{w}_k - w)^T F(w),
\end{aligned}$$

where the equality holds since for any  $w_1$  and  $w_2$ ,

$$(w_1 - w_2)^T (F(w_1) - F(w_2)) = 0,$$

and the inequality follows from the convexity of  $\theta$ . Now summing both sides of (22) from 0 to  $k$  and then taking the average, and using (23), the assertion of this theorem follows directly.  $\square$

### Proof of Corollary 1

*Proof.* In (9), let  $w = (x^*, y^*, \lambda)$ , and  $k = N$ , where  $\lambda = \lambda^* + e$  and  $e$  is a vector satisfying  $-e^T (A\bar{x}_N + B\bar{y}_N - b) = \|A\bar{x}_N + B\bar{y}_N - b\|$ . Obviously,  $\|e\| = 1$ . Then the left hand side of (9) is

$$\theta(\bar{u}_N) - \theta(u^*) - (\lambda^*)^T (A\bar{x}_N + B\bar{y}_N - b) + \|A\bar{x}_N + B\bar{y}_N - b\|. \quad (24)$$

Such a result is attributed to

$$\begin{aligned}
 & (\bar{w}_N - w)^T F(w) \\
 &= (\bar{x}_N - x^*)^T (-A^T \lambda) + (\bar{y}_N - y^*)^T (-B^T \lambda) + (\bar{\lambda}_N - \lambda)^T (Ax^* + By^* - b) \\
 &= \lambda^T (Ax^* + By^* - b) - (\lambda^T (A\bar{x}_N + B\bar{y}_N - b)) \\
 &= -(\lambda^*)^T (A\bar{x}_N + B\bar{y}_N - b) + \|A\bar{x}_N + B\bar{y}_N - b\|,
 \end{aligned}$$

where the first equality follows from the definition of  $F$ , and the second and last equalities hold due to  $Ax^* + By^* - b = 0$  and the choice of  $\lambda$ . On the other hand, substituting  $w = \bar{w}_N$  into the variational inequality associated with (1), we get

$$\theta(\bar{u}_N) - \theta(u^*) - (\lambda^*)^T (A\bar{x}_N + B\bar{y}_N - b) \geq 0. \quad (25)$$

Combining (24) and (25), we obtain that the left hand side of (9) is no less than  $\|A\bar{x}_N + B\bar{y}_N - b\|$  when letting  $w = (x^*, y^*, \lambda)$  and  $k = N$ . Hence,

$$\begin{aligned}
 & \mathbb{E} [\|A\bar{x}_N + B\bar{y}_N - b\|] \\
 & \leq \frac{1}{2(N+1)} \sum_{t=0}^N \left( \|w^t - w^*\|_{H_t}^2 - \|w^{t+1} - w^*\|_{H_t}^2 \right) + \frac{1}{N+1} \sum_{t=0}^N \frac{\xi_t}{2} \sigma^2 \\
 & \leq \frac{1}{2(N+1)} \left( M \|x^0 - x^*\|_{G_{1,0}}^2 + \|y^0 - y^*\|_{\frac{\beta}{\alpha} B^T B + G_2}^2 + \frac{2}{\beta\alpha} \left( \|\lambda^0 - \lambda^*\|^2 + 1 \right) \right) \\
 & \quad + \frac{1}{2\sqrt{N}} \left( \sigma^2 + \|x^0 - x^*\|_{G_{1,0}}^2 \right) + \frac{1-\alpha}{(N+1)\alpha} (\lambda^0 - \lambda^*)^T B (y^0 - y^*).
 \end{aligned} \quad (26)$$

where in the first inequality we use  $\mathbb{E}[\delta^k] = 0$  and  $\mathbb{E}[\|\delta^k\|^2] \leq \sigma^2$ . The first part of this corollary is proved. Next we prove the second part. Substituting  $w = \bar{w}_N$  into the variational inequality associated with (1), we get

$$\begin{aligned}
 & \theta(\bar{u}_N) - \theta(u^*) + (\bar{w}_N - w^*)^T F(w^*) \\
 &= \theta(\bar{u}_N) - \theta(u^*) - (\lambda^*)^T (A\bar{x}_N + B\bar{y}_N - b) \\
 &\geq \theta(\bar{u}_N) - \theta(u^*) - \|\lambda^*\| \|A\bar{x}_N + B\bar{y}_N - b\|,
 \end{aligned}$$

i.e.,

$$\theta(\bar{u}_N) - \theta(u^*) \leq \theta(\bar{u}_N) - \theta(u^*) + (\bar{w}_N - w^*)^T F(w^*) + \|\lambda^*\| \|A\bar{x}_N + B\bar{y}_N - b\|. \quad (27)$$

Taking expectation on both sides of (27) to complete the proof.  $\square$

**Proof of Corollary 2**

*Proof.* The proof of this corollary is almost similar to the corollary 1, except for estimating  $\mathbb{E} [\|A\bar{x}_k + B\bar{y}_k - b\|]$ .

$$\begin{aligned}
 & \mathbb{E} [\|A\bar{x}_k + B\bar{y}_k - b\|] \\
 \leq & \frac{1}{2(k+1)} \sum_{t=0}^k \left( \|w^t - w^*\|_{H_t}^2 - \|w^{t+1} - w^*\|_{H_t}^2 \right) + \frac{1}{k+1} \sum_{t=0}^k \frac{\alpha_t}{2} \sigma^2 \\
 \leq & \frac{1}{2(k+1)} \left( \frac{1}{\eta_0} \|w^0 - w^*\|_{G_{1,0}}^2 + \sum_{i=0}^{k-1} \left( \frac{1}{\eta_{i+1}} - \frac{1}{\eta_i} \right) \mathbb{E} \|w^{i+1} - w^*\|_{G_{1,i}}^2 \right. \\
 & \left. - \frac{1}{\eta_k} \mathbb{E} \|w^{k+1} - w^*\|_{G_{1,k}}^2 + \|y^0 - y^*\|_{\frac{\beta}{\alpha} B^T B + G_2}^2 + \frac{2}{\beta\alpha} \left( \|\lambda^0 - \lambda^*\|^2 + 1 \right) \right) \\
 & + \frac{1}{k+1} \sum_{t=0}^k \frac{1}{2\sqrt{t}} \sigma^2 + \frac{(1-\alpha)(\|\lambda^*\|+1)}{(k+1)\alpha} (\lambda^0 - \lambda^*)^T B (y^0 - y^*) \\
 \leq & \frac{1}{2(k+1)} \left( \frac{R^2}{\eta_0} + \sum_{i=0}^{k-1} \left( \frac{1}{\eta_{i+1}} - \frac{1}{\eta_i} \right) R^2 + \frac{2}{\beta\alpha} \left( \|\lambda^0 - \lambda^*\|^2 + 1 \right) \right. \\
 & \left. + \|y^0 - y^*\|_{\frac{\beta}{\alpha} B^T B + G_2}^2 \right) + \frac{1}{\sqrt{k}} \sigma^2 + \frac{(1-\alpha)(\|\lambda^*\|+1)}{(k+1)\alpha} (\lambda^0 - \lambda^*)^T B (y^0 - y^*) \\
 \leq & \frac{1}{2(k+1)} \left( \|y^0 - y^*\|_{\frac{\beta}{\alpha} B^T B + G_2}^2 + \frac{2}{\beta\alpha} \left( \|\lambda^0 - \lambda^*\|^2 + 1 \right) + MR^2 \right) \\
 & + \frac{1}{2\sqrt{k}} (2\sigma^2 + R^2) + \frac{(1-\alpha)(\|\lambda^*\|+1)}{(k+1)\alpha} (\lambda^0 - \lambda^*)^T B (y^0 - y^*).
 \end{aligned}$$

**Proof of Theorem 2**

*Proof.* First, similar to the proof of lemma 2, using the  $\mu$ -strong convexity of  $f$ , we conclude that for any  $w \in \Omega$

$$\begin{aligned}
 & \theta(u) - \theta(\tilde{u}^k) + (w - \tilde{w}^k)^T F(\tilde{w}^k) \\
 \geq & (w - \tilde{w}^k)^T Q_k (w^k - \tilde{w}^k) - (x - \tilde{x}^k)^T \delta^k - \frac{L}{2} \|x^k - \tilde{x}^k\|^2 + \frac{\mu}{2} \|x - x^k\|^2,
 \end{aligned} \tag{28}$$

where  $Q_k$  is defined in (7). Then using the result in lemma 3,

$$\begin{aligned}
 & (w - \tilde{w}^k)^T Q_k (w^k - \tilde{w}^k) \\
 = & \frac{1}{2} \left( \|w - w^{k+1}\|_{H_k}^2 - \|w - w^k\|_{H_k}^2 \right) + \frac{1}{2\eta_k} \|x^k - \tilde{x}^k\|_{G_{1,k}}^2 + \frac{1}{2} \|y^k - \tilde{y}^k\|_{G_2}^2 \\
 & - \frac{\alpha - 2}{2\beta} \|\lambda^k - \tilde{\lambda}^k\|^2.
 \end{aligned} \tag{29}$$

Combining (28) and (29), we get

$$\begin{aligned} & \theta(\tilde{u}^t) - \theta(u) + (\tilde{w}^t - w)^T F(\tilde{w}^t) \\ & \leq \frac{1}{2} \left( \|w^t - w\|_{H_t}^2 - \|w^{t+1} - w\|_{H_t}^2 - \mu \|x^t - x\|^2 \right) + (x - x^t)^T \delta^t + \frac{\alpha_t}{2} \|\delta^t\|^2. \end{aligned} \quad (30)$$

Now using (23) and (30), we have

$$\begin{aligned} & \theta(\bar{u}_k) - \theta(u) + (\bar{w}_k - w)^T F(w) \\ & \leq \frac{1}{k+1} \sum_{t=0}^k \theta(\tilde{u}^t) - \theta(u) + (\tilde{w}^t - w)^T F(\tilde{w}^t) \\ & \leq \frac{1}{2(k+1)} \sum_{t=0}^k \left( \frac{1}{\eta t} \|x^t - x\|_{G_{1,t}}^2 - \frac{1}{\eta t} \|x^{t+1} - x\|_{G_{1,t}}^2 - \mu \|x^t - x\|^2 \right) + \frac{1}{k+1} \sum_{t=0}^k \frac{\alpha_t}{2} \|\delta^t\|^2 \\ & \quad + \frac{1}{2(k+1)} \left( \|y^0 - y\|_{\frac{\beta}{\alpha} B^T B + G_2}^2 + \frac{1}{\beta\alpha} \|\lambda^0 - \lambda\|^2 \right) + \frac{1}{k+1} \sum_{t=0}^k (x - x^t)^T \delta^t \\ & \quad + \frac{1-\alpha}{(k+1)\alpha} (\lambda^0 - \lambda)^T B (y^0 - y) \\ & \leq \frac{1}{2(k+1)} \sum_{t=0}^k \left( (\mu t + M) \|x^t - x\|^2 - (\mu(t+1) + M) \|x^{t+1} - x\|^2 \right) \\ & \quad + \frac{1}{2(k+1)} \left( \|y^0 - y\|_{\frac{\beta}{\alpha} B^T B + G_2}^2 + \frac{1}{\beta\alpha} \|\lambda^0 - \lambda\|^2 + \|x^0 - x\|_{\tau I_n}^2 - \beta A^T A \right) \\ & \quad + \frac{1}{k+1} \sum_{t=0}^k (x - x^t)^T \delta^t + \frac{1}{k+1} \sum_{t=0}^k \frac{\alpha t}{2} \|\delta^t\|^2 + \frac{1-\alpha}{(k+1)\alpha} (\lambda^0 - \lambda)^T B (y^0 - y) \\ & \leq \frac{1}{2(k+1)} \left( \|x^0 - x\|_{(\tau+M)I_n - \beta A^T A}^2 + \|y^0 - y\|_{\frac{\beta}{\alpha} B^T B + G_2}^2 + \frac{1}{\beta\alpha} \|\lambda^0 - \lambda\|^2 \right) \\ & \quad + \frac{1}{k+1} \sum_{t=0}^k (x - x^t)^T \delta^t + \frac{1}{k+1} \sum_{t=0}^k \frac{\alpha}{2} \|\delta^t\|^2 + \frac{1-\alpha}{(k+1)\alpha} (\lambda^0 - \lambda)^T B (y^0 - y). \end{aligned} \quad (31)$$

Finally, taking expectation on both sides of (29) and following the proof for getting (26) and (27), we obtain

$$\begin{aligned} & \mathbb{E}[\|A\bar{x}_k + B\bar{y}_k - b\|] \\ & \leq \frac{1}{2(k+1)} \left( \|x^0 - x^*\|_{(\tau+M)I_n - \beta A^T A}^2 + \|y^0 - y^*\|_{\frac{\beta}{\alpha} B^T B + G_2}^2 + \frac{2}{\beta\alpha} (\|\lambda^0 - \lambda^*\|^2 + 1) \right) \\ & \quad + \frac{\sigma^2}{2\mu(k+1)} (1 + \ln(k+1)) + \frac{1-\alpha}{(k+1)\alpha} (\lambda^0 - \lambda^*)^T B (y^0 - y^*) \end{aligned}$$

and

$$\theta(\bar{u}_k) - \theta(u^*) \leq \theta(\bar{u}_k) - \theta(u^*) + (\bar{w}_k - w^*)^T F(w^*) + \|\lambda^*\| \|A\bar{x}_k + B\bar{y}_k - b\|.$$

Therefore, this theorem is proved.  $\square$

### Proof of Theorem 3

*Proof.* Since  $(x^*, y^*, \lambda^*)$  is a solution of (1), we have

$$A^T \lambda^* = \nabla \theta_1(x^*) \text{ and } B^T \lambda^* \in \partial \theta_2(y^*).$$

Hence, since  $\theta_1$  is strongly convex and  $\theta_2$  is convex, we have

$$\theta_1(\bar{x}_k) \geq \theta_1(x^*) + (\lambda^*)^T (A\bar{x}_k - Ax^*) + \frac{\mu}{2} \|\bar{x}_k - x^*\|^2 \tag{32}$$

and

$$\theta_2(\bar{y}_k) \geq \theta_2(y^*) + (\lambda^*)^T (B\bar{y}_k - By^*). \tag{33}$$

Adding up (32) and (33), we get

$$\theta(\bar{u}_k) \geq \theta(u^*) + (\lambda^*)^T (A\bar{x}_k + B\bar{y}_k - b) + \frac{\mu}{2} \|\bar{x}_k - x^*\|^2.$$

Taking expectation gives

$$\begin{aligned} \|\bar{x}_k - x^*\|^2 &\leq \frac{2}{\mu} \left( \theta(\bar{u}_k) - \theta(u^*) - (\lambda^*)^T (A\bar{x}_k + B\bar{y}_k - b) \right) \\ &\leq \frac{2}{\mu} (\theta(\bar{u}_k) - \theta(u^*) + \|\lambda^*\| \|A\bar{x}_k + B\bar{y}_k - b\|). \end{aligned} \tag{34}$$

On the other hand,

$$\begin{aligned} \|A\bar{x}_k + B\bar{y}_k - b\| &= \|A(\bar{x}_k - x^*) + B(\bar{y}_k - y^*)\| \\ &\geq \|B(\bar{y}_k - y^*)\| - \|A\| \|\bar{x}_k - x^*\|, \end{aligned}$$

this implies  $\|B(\bar{y}_k - y^*)\|^2 \leq 2\|A\|^2 \|\bar{x}_k - x^*\|^2 + 2\|A\bar{x}_k + B\bar{y}_k - b\|^2$  and hence

$$\|\bar{y}_k - y^*\|^2 \leq \frac{2\|A\|^2}{s} \|\bar{x}_k - x^*\|^2 + \frac{2}{s} \|A\bar{x}_k + B\bar{y}_k - b\|^2. \tag{35}$$

Adding (34) and (35), and taking expectation imply

$$\begin{aligned} \mathbb{E} \left[ \|\bar{x}_k - x^*\|^2 + \|\bar{y}_k - y^*\|^2 \right] &\leq \left( \frac{2}{\mu} + \frac{4\|A\|^2}{\mu s} \right) (\mathbb{E} [\theta(\bar{u}_k) - \theta(u^*)]) \\ &\quad + \|\lambda^*\| \mathbb{E} [\|A\bar{x}_k + B\bar{y}_k - b\|] + \frac{2}{s} \mathbb{E} [\|A\bar{x}_k + B\bar{y}_k - b\|^2]. \end{aligned} \tag{36}$$

The remaining task is to estimate  $\mathbb{E} [\|A\bar{x}_k + B\bar{y}_k - b\|^2]$ .

In (9), let  $w = (x^*, y^*, \lambda)$ , where  $\lambda = \lambda^* + e$ , and  $e$  is a vector satisfying  $-e^T (A\bar{x}_k + B\bar{y}_k - b) = \|A\bar{x}_k + B\bar{y}_k - b\|^2$ . Then, similar to the proof idea of getting (26), we get

$$\begin{aligned} &\mathbb{E} \left[ \|A\bar{x}_k + B\bar{y}_k - b\|^2 \right] \\ &\leq \frac{1}{2(k+1)} \left( M \|x^0 - x^*\|_{G_{1,0}}^2 + \|y^0 - y^*\|_{\frac{\alpha}{\beta} B^T B + G_2}^2 + \frac{2}{\beta\alpha} \|\lambda^0 - \lambda^*\|^2 \right) \\ &\quad + \frac{1}{2\sqrt{k}} \left( \sigma^2 + \|x^0 - x^*\|_{G_{1,0}}^2 \right) + \frac{1}{\beta\alpha(k+1)} \mathbb{E} \left[ \|A\bar{x}_k + B\bar{y}_k - b\|^2 \right] \\ &\quad + \frac{1-\alpha}{(k+1)\alpha} (\lambda^0 - \lambda^*)^T B (y^0 - y^*) \end{aligned} \tag{37}$$

Arranging this inequality, we obtain the desired bound and the proof is completed.  $\square$

## References

1. Glowinski, R., Marroco, A.: Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de Dirichlet non linéaires. *Revue française d'automatique, informatique, recherche opérationnelle. Analyse numérique* **9**(R2), 41–76 (1975)
2. Gabay, D., Mercier, B.: A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Comput. Math. Appl.* **2**(1), 17–40 (1976)
3. Glowinski, R.: On alternating direction methods of multipliers: a historical perspective. In: Fitzgibbon, W., Kuznetsov, Y.A., Neittaanmäki, P., Pironneau, O. (eds.) *Modeling, Simulation and Optimization for Science and Technology*. CMAS, vol. 34, pp. 59–82. Springer, Dordrecht (2014). [https://doi.org/10.1007/978-94-017-9054-3\\_4](https://doi.org/10.1007/978-94-017-9054-3_4)
4. Eckstein, J., Bertsekas, D.P.: On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Math. Program.* **55**(1), 293–318 (1992)
5. He, B., Yuan, X.: On the  $O(1/n)$  convergence rate of the Douglas-Rachford alternating direction method. *SIAM J. Numer. Anal.* **50**(2), 700–709 (2012)
6. Monteiro, R.D.C., Svaiter, B.F.: Iteration-complexity of block-decomposition algorithms and the alternating direction method of multipliers. *SIAM J. Optim.* **23**(1), 475–507 (2013)
7. He, B., Yuan, X.: On non-ergodic convergence rate of Douglas-Rachford alternating direction method of multipliers. *Numer. Math.* **130**(3), 567–577 (2015)
8. Deng, W., Yin, W.: On the global and linear convergence of the generalized alternating direction method of multipliers. *J. Sci. Comput.* **66**(3), 889–916 (2016)
9. Yang, W.H., Han, D.: Linear convergence of the alternating direction method of multipliers for a class of convex optimization problems. *SIAM J. Numer. Anal.* **54**(2), 625–640 (2016)
10. Han, D., Sun, D., Zhang, L.: Linear rate convergence of the alternating direction method of multipliers for convex composite programming. *Math. Oper. Res.* **43**(2), 622–637 (2018)
11. Li, G., Pong, T.K.: Global convergence of splitting methods for nonconvex composite optimization. *SIAM J. Optim.* **25**(4), 2434–2460 (2015)
12. Wang, Y., Yin, W., Zeng, J.: Global convergence of ADMM in nonconvex nonsmooth optimization. *J. Sci. Comput.* **78**(1), 29–63 (2019)
13. Jiang, B., Lin, T., Ma, S., et al.: Structured nonconvex and nonsmooth optimization: algorithms and iteration complexity analysis. *Comput. Optim. Appl.* **72**(1), 115–157 (2019)
14. Zhang, J., Luo, Z.Q.: A proximal alternating direction method of multiplier for linearly constrained nonconvex minimization. *SIAM J. Optim.* **30**(3), 2272–2302 (2020)
15. Robbins, H., Monro, S.: A stochastic approximation method. *Ann. Math. Stat.* **22**(3), 400–407 (1951)
16. Nemirovski, A., Juditsky, A., Lan, G., et al.: Robust stochastic approximation approach to stochastic programming. *SIAM J. Optim.* **19**(4), 1574–1609 (2009)
17. Ghadimi, S., Lan, G.: Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM J. Optim.* **23**(4), 2341–2368 (2013)
18. Ghadimi, S., Lan, G.: Accelerated gradient methods for nonconvex nonlinear and stochastic programming. *Math. Program.* **156**(1), 59–99 (2016)

19. Lan, G.: An optimal method for stochastic composite optimization. *Math. Program.* **133**(1), 365–397 (2012)
20. Ghadimi, S., Lan, G., Zhang, H.: Mini-batch stochastic approximation methods for nonconvex stochastic composite optimization. *Math. Program.* **155**(1), 267–305 (2016)
21. Ouyang, H., He, N., Tran, L., et al.: Stochastic alternating direction method of multipliers. In: *Proceedings of the 30th International Conference on Machine Learning*, pp. 80–88. PMLR, Atlanta (2013)
22. Suzuki, T.: Dual averaging and proximal gradient descent for online alternating direction multiplier method. In: *Proceedings of the 30th International Conference on Machine Learning*, pp. 392–400. PMLR, Atlanta (2013)
23. Suzuki, T.: Stochastic dual coordinate ascent with alternating direction method of multipliers. In: *Proceedings of the 31th International Conference on Machine Learning*, pp. 736–744. PMLR, Beijing (2014)
24. Zhao, P., Yang, J., Zhang, T., et al.: Adaptive stochastic alternating direction method of multipliers. In: *Proceedings of the 32th International Conference on Machine Learning*, pp. 69–77. PMLR, Lille (2014)
25. Gao, X., Jiang, B., Zhang, S.: On the information-adaptive variants of the ADMM: an iteration complexity perspective. *J. Sci. Comput.* **76**(1), 327–363 (2018)
26. Fang, E.X., He, B., Liu, H., Yuan, X.: Generalized alternating direction method of multipliers: new theoretical insights and applications. *Math. Program. Comput.* **7**(2), 149–187 (2015). <https://doi.org/10.1007/s12532-015-0078-2>





# A 3/4 Differential Approximation Algorithm for Traveling Salesman Problem

Yuki Amano<sup>(✉)</sup> and Kazuhisa Makino

Research Institute for Mathematical Sciences, Kyoto University, Kyoto, Japan  
{ukiamano,makino}@kurims.kyoto-u.ac.jp

**Abstract.** In this paper, we consider differential approximability of the traveling salesman problem (TSP). The differential approximation ratio was proposed by Demange and Paschos in 1996 as an approximation criterion that is invariant under affine transformation of the objective function. We show that TSP is 3/4-differential approximable, which improves the currently best known bound  $3/4 - O(1/n)$  due to Escoffier and Monnot in 2008, where  $n$  denotes the number of vertices in the given graph.

**Keywords:** Approximation algorithms · Differential approximation · Traveling salesman problem

## 1 Introduction

The traveling salesman problem (TSP) is to find a shortest *Hamiltonian cycle* in a given complete graph with edge lengths, where a cycle is called *Hamiltonian* (also called a *tour*) if it visits every vertex exactly once. TSP is one of the most fundamental NP-hard optimization problems in operations research and computer science, and has been intensively studied from both practical and theoretical viewpoints [8, 22, 24, 26]. It has a number of applications such as planning, logistics, and the manufacture of microchips [6, 12]. Because of its importance, many heuristics and exact algorithms have been proposed [5, 14, 16, 17]. From the viewpoint of computational complexity, TSP is NP-hard, even in the Euclidean case, implying NP-hardness of the metric case. It is known that metric TSP is approximable with factor  $(3/2 - \epsilon)$  for some  $\epsilon > 10^{-36}$  [15], and inapproximable with factor at most 117/116, unless  $P = NP$  [7]. Euclidean TSP admits a polynomial-time approximation scheme (PTAS), if the dimension of the Euclidean space is bounded by a constant [2, 18]. We note that the approximation factors (i.e., ratios) above are widely used to analyze approximation algorithms.

Let  $\Pi$  be an optimization problem, and let  $I$  be an instance of  $\Pi$ . We denote by  $\text{opt}(I)$  the value of an optimal solution to  $I$ . For an approximation algorithm  $A$  for  $\Pi$ , we denote by  $\text{apx}_A(I)$  the value of the approximate solution computed by  $A$  for the instance  $I$ . Let

$$r_A(I) = \text{apx}_A(I) / \text{opt}(I),$$

and define the *standard approximation ratio* of  $A$  by  $\sup_{I \in \Pi} r_A(I)$ , where we assume that  $\Pi$  is a minimization problem. Although the standard approximation ratio is well-studied and an important concept in algorithm theory, it is not invariant under affine transformation of the objective function. Namely, if the objective function  $f(x)$  is replaced by  $a + bf(x)$  for some constant  $a$  and  $b$ , which might depend on the instance  $I$ , the standard ratio is not preserved. For example, the vertex cover problem and the independent set problem have affinely dependent objective functions. However they have different characteristics in the standard approximation ratio. The vertex cover problem is 2-approximable [23], while the independent set problem is inapproximable within  $O(n^{1-\epsilon})$  for any  $\epsilon > 0$ , unless  $P = NP$  [10], where  $n$  denotes the number of vertices in a given graph. In order to remedy to this phenomenon, Demange and Paschos [9] proposed the *differential approximation ratio* defined by  $\inf_{I \in \Pi} \rho_A(I)$ , where

$$\rho_A(I) = \frac{\text{wor}(I) - \text{apx}_A(I)}{\text{wor}(I) - \text{opt}(I)}$$

and  $\text{wor}(I)$  denotes the value of a worst solution to  $I$ . Note that for any instance  $I$  of  $\Pi$

$$\text{apx}_A(I) = \rho_A(I)\text{opt}(I) + (1 - \rho_A(I))\text{wor}(I).$$

Thus we have  $0 \leq \rho_A(I) \leq 1$  and the larger  $\rho_A(I)$  implies the better approximation for the instance  $I$ . This means that differential approximation ratio makes use of not only the optimal value but also the worst value. Moreover, by definition, the differential approximation ratio remains invariant under affine transformation of the objective function. Ausiello and Paschos [4] also mention that differential approximability creates the complexity classes corresponding to the ones for standard approximability. For these reasons, differential approximation has recently attracted much attention in approximation algorithm and a number of optimization problems have been studied from the viewpoint of differential approximation; see e.g., [3, 4]. It is known [20] that TSP, metric TSP, max TSP, and max metric TSP are affinely equivalent, i.e., their objective functions are transferred to each other by affine transformations, where max TSP is the problem to find a longest Hamiltonian cycle and max metric TSP is max TSP, in which the input weighted graph satisfies the metric condition. Therefore, these problems have identical differential approximation ratio.

Hassin and Khuller [13] first studied differential approximability of TSP, and showed that it is  $2/3$ -differential approximable. Escoffier and Monnot [11] improved it to  $3/4 - O(1/n)$ , where  $n$  denotes the number of vertices of the given graph. Monnot et al. [19, 21] showed that TSP is  $3/4$ -differential approximable if each edge length is restricted to one or two.

In this paper, we show that TSP is  $3/4$ -differential approximable, which improves the currently best known results [11, 19, 21]. Our algorithm is based on an idea by Escoffier and Monnot [11] for the case in which a given graph  $G$  has an even number of vertices and a triangle (i.e., cycle with 3 edges) is contained in a minimum weighted 2-factor of  $G$ . Their algorithm first computes minimum weighted 1- and 2-factors of a given graph, modifies them to four path covers

$P_i$  (for  $i = 1, \dots, 4$ ), and then extends each path cover  $P_i$  to a tour by adding edge set  $F_i$  to it in such a way that at least one of the four tours guarantees 3/4-differential approximation ratio. The definitions of factor and path cover can be found in Sect. 2. We generalize their idea to the graphs with an even number of vertices. Note that  $\bigcup_{i=1, \dots, 4} F_i$  in their algorithm always forms a tour, while in general it does not. We show that there exists a way to construct path covers such that the length of  $\bigcup_{i=1, \dots, 4} F_i$  is at most the worst tour length. Our algorithm for odd case is much more involved. For each path with three edges, we first construct a 2-factor and two path covers of a given graph that has the minimum length among all these that completely and partially contain the path, modify them to eight path covers, and then extend each path cover to a tour, in such a way that at least one of the eight tours guarantees 3/4-differential approximation ratio.

The rest of the paper is organized as follows. In Sect. 2, we define basic concepts of graphs and discuss some properties on 2-matchings, which will be used in the subsequent sections. In Sects. 3 and 4, we provide approximation algorithms for TSP in which a given graph  $G$  has even and odd numbers of vertices, respectively.

Due to space constraints, most of the proofs are omitted in this paper, which can be found in [1].

## 2 Definitions and Basic Properties

Let  $G = (V, E)$  be an undirected graph, where  $n$  and  $m$  denote the number of vertices and edges in  $G$ , respectively. In this paper, we assume that a given graph  $G$  of TSP is complete, i.e.,  $m = \binom{n}{2}$ , and it has an edge length function  $\ell : E \rightarrow \mathbb{R}_+$ , where  $\mathbb{R}_+$  denotes the set of nonnegative reals. For a set  $F \subseteq E$ , let  $V(F)$  denotes the set of vertices with incident edges in  $F$ , i.e.,  $V(F) = \{v \in V \mid \exists (v, w) \in F\}$ . A set  $F \subseteq E$  is called *spanning* if  $V(F) = V$ , and *acyclic* if  $F$  contains no cycle. For a positive integer  $k$ , a set  $F \subseteq E$  is called a *k-matching* (resp., *k-factor*) if each vertex has at most (resp., exactly)  $k$  incident edges in  $F$ . Here 1-matching is simply called a *matching*. Note that an acyclic 2-matching  $F$  corresponds to a family of vertex-disjoint paths denoted by  $\mathcal{P}(F) \subseteq 2^F$ . A 2-matching is called a *path cover* if it is spanning and acyclic. For a set  $F \subseteq E$ ,  $V_1(F)$  and  $V_2(F)$  respectively denote the sets of vertices with one and two incident edges in  $F$ . For a set  $F \subseteq E$  and a vertex  $v \in V$ , let  $\delta_F(v) = \{e \in F \mid e \text{ is incident to } v\}$ .

Let us now define valid pairs of spanning 2-matchings.

**Definition 1.** A pair of spanning 2-matchings  $(S, T)$  is called valid if it satisfies the following three conditions:

- (i)  $T$  is acyclic (i.e., a path cover).
- (ii)  $\delta_S(v) = \delta_T(v)$  for any  $v \in V_2(S) \cap V_2(T)$ . (1)
- (iii)  $V(C) \neq V(P)$  for any cycle  $C \subseteq S$  and any path  $P \in \mathcal{P}(T)$ . (2)

The following lemma plays a crucial role in our approximation algorithms.

**Lemma 2.** *Let  $(S, T)$  be a valid pair of spanning 2-matchings. If  $S$  contains a cycle  $C$ , then  $C \setminus T$  contains such two edges  $e_1$  and  $e_2$  that  $S_i = S \setminus \{e_i\}$  and  $T_i = T \cup \{e_i\}$  for  $i = 1, 2$  satisfy the following two conditions:*

$$(i) \quad (S_i, T_i) \text{ is a valid pair of spanning 2-matchings for } i = 1, 2. \tag{3}$$

$$(ii) \quad \mathcal{P}(T) \text{ contains a path } P \text{ such that } P \cup \{e_1\} \text{ and } P \cup \{e_2\} \text{ are both paths.} \tag{4}$$

Note that  $(S_1, T_1)$  and  $(S_2, T_2)$  in Lemma 2 satisfy

$$V_1(S_i) \cup V_1(T_i) = V_1(S) \cup V_1(T) \tag{5}$$

and  $V_1(S_i) \cap V_1(T_i) = V_1(S) \cap V_1(T)$  for  $i = 1, 2,$

$$S_i \cup T_i = S \cup T \text{ and } S_i \cap T_i = S \cap T \text{ for } i = 1, 2, \tag{6}$$

where (6) immediately implies

$$\ell(S_i) + \ell(T_i) = \ell(S) + \ell(T) \text{ for } i = 1, 2, \tag{7}$$

where  $\ell(F) = \sum_{e \in F} \ell(e)$  for any set  $F \subseteq E$ .

Figure 1 shows a valid pair of spanning 2-matchings and two edges  $e_c$  and  $e_d$  in  $C \setminus T$  that satisfy the two conditions of Lemma 2.

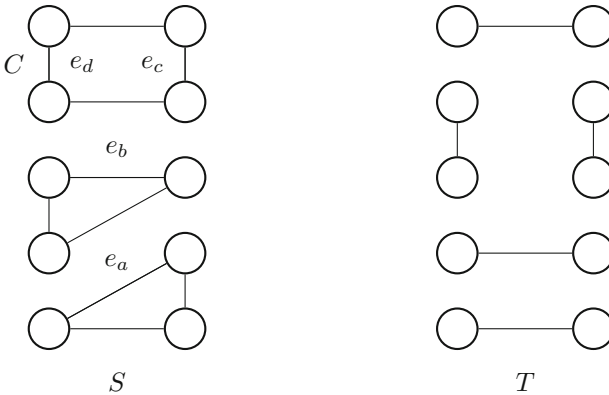


Fig. 1. A valid pair  $(S, T)$  of spanning 2-matchings.

### 3 Approximation for Even Instances

In this section, we consider the case in which a given graph has an even number of vertices. As explained in the introduction,  $3/4$ -differential approximability is known for this case [11]. However their algorithm consists of two different

---

**Procedure FourPathCovers**( $S, T$ )

/\*( $S, T$ ) is a valid pair of spanning 2-matchings such that  $S$  has a cycle. The procedure returns 4 path covers  $S_1, S_2, T_1,$  and  $T_2$  that satisfies (5), (6), and (9).\*/

---

**if**  $S$  has exactly one cycle **then**

    Take two edges  $e_1$  and  $e_2$  as in Lemma 2.

**return**  $S_1 = S \setminus \{e_1\}, T_1 = T \cup \{e_1\}, S_2 = S \setminus \{e_2\},$  and  $T_2 = T \cup \{e_2\}$

**else** /\*  $S$  has at least two cycles. \*/

    Take an edge  $e_1$  in Lemma 2.

**return** **FourPathCovers**( $S \setminus \{e_1\}, T \cup \{e_1\}$ )

**end if**

---

subroutines in this case. In this section we propose a unified algorithm, which can further be extended to the odd case. Our algorithm first constructs four path covers from minimum weighted 1- and 2-factors of a given graph  $G$ , and then extends each path cover to a tour in such a way that at least one of the tours guarantees 3/4-differential approximation ratio.

Let us first describe procedure **FourPathCovers**. Let  $(S, T)$  be a valid pair of spanning 2-matchings of  $(G, \ell)$  such that  $S$  is a 2-factor. The procedure computes from  $(S, T)$  four path covers  $S_1, S_2, T_1,$  and  $T_2$  that satisfies (5), (6),  $V_1(S_i)$  and  $V_1(T_i)$  form a partition of  $V_1(T)$  for  $i = 1, 2,$  i.e.,

$$V_1(S_i) \cup V_1(T_i) = V_1(T) \text{ and } V_1(S_i) \cap V_1(T_i) = \emptyset \text{ for } i = 1, 2, \tag{8}$$

and

there exist  $e_1, e_2 \in E$  and  $P \in \mathcal{P}(T_1 \cap T_2)$  such that

$$T_1 \setminus T_2 = \{e_1\}, T_2 \setminus T_1 = \{e_2\}, P \cup \{e_1\} \in \mathcal{P}(T_1), \text{ and } P \cup \{e_2\} \in \mathcal{P}(T_2). \tag{9}$$

**Lemma 3.** *For a graph  $G = (V, E)$ , let  $(S, T)$  be a valid pair of spanning 2-matchings such that  $S$  has a cycle. Then **Procedure FourPathCovers** returns four path covers  $S_1, S_2, T_1,$  and  $T_2$  that satisfy (5), (6), and (9). Furthermore, if  $S$  is a 2-factor of  $G$ , then the four path covers satisfy (8).*

Let  $S$  and  $T$  be 2- and 1- factors of  $G$ , respectively. Then  $V_1(T) = V$  and  $(S, T)$  is a valid pair of spanning 2-matchings. Note that our algorithm explained later makes use of minimum weighted 2-factor  $S$  and 1-factor  $T$  of  $(G, \ell)$  that can be computed from  $(G, \ell)$  in polynomial time [25]. We assume that  $S$  is not a tour of  $G$ , i.e.,  $S$  contains at least two cycles, since otherwise,  $S$  itself is an optimal tour. Let  $S_1, S_2, T_1,$  and  $T_2$  be path covers returned by **Procedure FourPathCovers**( $S, T$ ).

Let us then show how to construct edge sets  $A_1, A_2, B_1,$  and  $B_2,$  such that

$$(i) \quad S_i \cup A_i \text{ is a tour (for } i = 1, 2), \tag{10}$$

$$(ii) \quad T_i \cup B_i \text{ is a tour (for } i = 1, 2), \text{ and} \tag{11}$$

$$(iii) \quad \ell(A_1) + \ell(A_2) + \ell(B_1) + \ell(B_2) \leq \text{wor}(G, \ell), \tag{12}$$

where  $\text{wor}(G, \ell)$  denotes the length of a longest tour of  $(G, \ell)$ .

Let  $e_1 = (p_1, p_2)$  and  $e_2 = (p_3, p_4)$  be edges in Lemma 3. Since  $e_1$  and  $e_2$  are chosen from a cycle  $C$  and satisfy (9), we can assume that  $p_1 \neq p_3, p_4$  and  $p_4 \neq p_1, p_2,$  where  $p_2 = p_3$  might hold. We note that  $\mathcal{P}(S_1) \setminus \mathcal{P}(S_2)$  consists of a  $(p_1, p_2)$ -path  $P_1 = C \setminus \{e_1\},$  and  $\mathcal{P}(S_2) \setminus \mathcal{P}(S_1)$  consists of a  $(p_3, p_4)$ -path  $P_2 = C \setminus \{e_2\}.$

Let us first construct  $A_1$  and  $A_2.$  Let  $Q_i (i = 1, \dots, k)$  denote vertex-disjoint  $(x_i, y_i)$ -paths such that  $\{Q_1, \dots, Q_k\} = \mathcal{P}(S_1) \cap \mathcal{P}(S_2)$  and  $x_1$  and  $y_1$  satisfy

$$\ell(p_2, x_1) + \ell(p_3, y_1) \leq \ell(p_2, y_1) + \ell(p_3, x_1). \tag{13}$$

Define  $A_1$  and  $A_2$  by

$$\begin{aligned} A_1 &= \{(p_2, x_1)\} \cup \{(y_i, x_{i+1}) \mid i = 1, \dots, k - 1\} \cup \{(y_k, p_1)\} \\ A_2 &= \{(p_3, y_1)\} \cup \{(x_i, y_{i+1}) \mid i = 1, \dots, k - 1\} \cup \{(x_k, p_4)\}. \end{aligned} \tag{14}$$

Then we have the following lemma.

**Lemma 4.** *Two sets  $A_1$  and  $A_2$  defined in (14) satisfy (10),*

- (i)  $V(A_i) = V_1(S_i)$  for  $i = 1, 2,$  and
- (ii)  $A_1 \cap A_2 = \emptyset$  and  $A_1 \cup A_2$  consists of
  1. a  $(p_1, p_4)$ -path if  $p_2 = p_3,$
  2. vertex-disjoint  $(p_1, p_3)$ - and  $(p_2, p_4)$ -paths if  $p_2 \neq p_3$  and  $k$  is odd,
  3. vertex-disjoint  $(p_1, p_2)$ - and  $(p_3, p_4)$ -paths if  $p_2 \neq p_3$  and  $k$  is even.

*Proof.* Note that  $\mathcal{P}(S_1) = \{Q_1, \dots, Q_k\} \cup \{P_1\}$  and  $\mathcal{P}(S_2) = \{Q_1, \dots, Q_k\} \cup \{P_2\}.$  Thus it follows from the definitions of  $A_1$  and  $A_2.$  □

Let us next construct  $B_1$  and  $B_2.$  Let  $O_i (i = 1, \dots, d)$  denote vertex-disjoint  $(z_i, w_i)$ -paths such that  $\{O_1, \dots, O_d\} = \mathcal{P}(T_1) \cap \mathcal{P}(T_2).$  Note that  $\mathcal{P}(T_1) \cap \mathcal{P}(T_2) = \emptyset$  (i.e.,  $d = 0$ ) might hold. We separately consider the following four cases, where we recall that  $p_1, p_2, p_3,$  and  $p_4$  are vertices such that  $e_1 = (p_1, p_2)$  and  $e_2 = (p_3, p_4)$  satisfy Lemma 3.

1.  $p_2 = p_3, p_1 \neq p_4,$  and  $\mathcal{P}(T_1 \cap T_2)$  contains a  $(p_1, p_4)$ -path.
2.  $p_2 = p_3, p_1 \neq p_4,$  and  $\mathcal{P}(T_1 \cap T_2)$  contain no  $(p_1, p_4)$ -path.
3.  $p_2 \neq p_3, p_1 \neq p_4,$  and  $\mathcal{P}(T_1 \cap T_2)$  contains  $(p_1, p_4)$ - and  $(p_2, p_3)$ -paths.
4.  $p_2 \neq p_3, p_1 \neq p_4,$  and  $\mathcal{P}(T_1 \cap T_2)$  contains a  $(p_2, p_3)$ -path and no  $(p_1, p_4)$ -path.

**Case 1:** Let  $R_1$  denotes a  $(p_1, p_4)$ -path in  $\mathcal{P}(T_1 \cap T_2)$ , and for some vertex  $q_2$ , let  $R_2$  denotes a  $(p_2, q_2)$ -path in  $\mathcal{P}(T_1 \cap T_2)$ . Note that such an  $R_2$  exists since  $T_1 \setminus \{e_1\} = T_2 \setminus \{e_2\}$ . Then, we have

$$\begin{aligned} \mathcal{P}(T_1) &= \{O_1, \dots, O_d\} \cup \{R_1 \cup \{e_1\} \cup R_2\} \\ \mathcal{P}(T_2) &= \{O_1, \dots, O_d\} \cup \{R_1 \cup \{e_2\} \cup R_2\}, \end{aligned}$$

where  $R_1 \cup \{e_1\} \cup R_2$  and  $R_1 \cup \{e_2\} \cup R_2$  are  $(p_4, q_2)$ - and  $(p_1, q_2)$ -paths, respectively. Define  $B_1$  and  $B_2$  by

$$\begin{aligned} B_1 &= \begin{cases} \{(q_2, p_4)\} & \text{if } d = 0 \\ \{(q_2, z_1)\} \cup \{(w_i, z_{i+1}) \mid i = 1, \dots, d - 1\} \cup \{(w_d, p_4)\} & \text{if } d \geq 1 \end{cases} \\ B_2 &= \begin{cases} \{(q_2, p_1)\} & \text{if } d = 0 \\ \{(q_2, w_1)\} \cup \{(z_i, w_{i+1}) \mid i = 1, \dots, d - 1\} \cup \{(z_d, p_1)\} & \text{if } d \geq 1. \end{cases} \end{aligned} \tag{15}$$

By definition, two edge sets  $B_1$  and  $B_2$  satisfy (11) and the following two statements.

- (i)  $V(B_i) = V_1(T_i)$  for  $i = 1, 2$ . (16)
- (ii)  $B_1 \cap B_2 = \emptyset$  and  $B_1 \cup B_2$  is a  $(p_1, p_4)$ -path. (17)

Similarly to Case 1, we also construct  $B_1$  and  $B_2$  for the other cases. These  $B_1$  and  $B_2$  satisfy the following lemma.

**Lemma 5.** *Let  $B_1$  and  $B_2$  be two edge sets defined as above. Then they satisfy (11), (16),  $B_1 \cap B_2 = \emptyset$ . Moreover,  $B_1 \cup B_2$  consists of*

1. a  $(p_1, p_4)$ -path if  $p_2 = p_3$ ,
2. vertex-disjoint  $(p_1, p_2)$ - and  $(p_3, p_4)$ -paths if  $p_2 \neq p_3$  and  $|\mathcal{P}(T_1)|$  is odd,
3. vertex-disjoint  $(p_1, p_3)$ - and  $(p_2, p_4)$ -paths if  $p_2 \neq p_3$  and  $|\mathcal{P}(T_1)|$  is even.

Furthermore,  $A_i$  and  $B_i$  ( $i = 1, 2$ ) satisfy the following properties.

**Lemma 6.** *Let  $A_1, A_2, B_1$ , and  $B_2$  be defined as above. Then they are all pairwise disjoint, and  $C = A_1 \cup A_2 \cup B_1 \cup B_2$  is a 2-factor, consisting of either one or two cycles. Furthermore, there exists a tour  $H$  of  $G$  such that  $\ell(H) \geq \ell(C)$ .*

We are now ready to describe our conclusion on the approximation algorithm.

**Theorem 7.** *For a complete graph  $G = (V, E)$  with an even number of vertices and an edge length function  $\ell : E \rightarrow \mathbb{R}_+$ , **Algorithm TourEven** computes a 3/4-differential approximate tour of  $(G, \ell)$  in polynomial time.*

*Proof.* We show that **Algorithm TourEven** outputs a 3/4-differential approximate tour  $T_{\text{apx}}$  in polynomial time. If a minimum weighted 2-factor  $S$  of  $(G, \ell)$  computed in the algorithm is a tour, then clearly  $T_{\text{apx}} = S$  is an optimal tour. On the other hand, if  $S$  is not a tour, then we have

$$\begin{aligned} 4\ell(T_{\text{apx}}) &\leq \ell(S_1 \cup A_1) + \ell(S_2 \cup A_2) + \ell(T_1 \cup B_1) + \ell(T_2 \cup B_2) \\ &= 2\ell(S) + \ell(T) + \ell(A_1 \cup A_2 \cup B_1 \cup B_2) \\ &\leq 3\text{opt}(G, \ell) + \text{wor}(G, \ell), \end{aligned}$$

---

**Algorithm. TourEven**

---

**Input:** A complete graph  $G = (V, E)$  with even  $|V|$ , and an edge length function  $\ell : E \rightarrow \mathbb{R}_+$ .

**Output:** A tour  $T_{\text{apx}}$  in  $G$ .

Compute minimum weighted 2-factor  $S$  and 1-factor  $T$  of  $(G, \ell)$ .

**if**  $S$  is a tour **then**

$T_{\text{apx}} := S$ .

**else**

$S_1, T_1, S_2, T_2 := \text{FourPathCovers}(S, T)$ .

Compute edge sets  $A_1$  and  $A_2$  defined in (14).

Compute edge sets  $B_1$  and  $B_2$ , which satisfy conditions of Lemma 5.

$\mathcal{T} := \{S_1 \cup A_1, S_2 \cup A_2, T_1 \cup B_1, T_2 \cup B_2\}$ .

$T_{\text{apx}} := \underset{T \in \mathcal{T}}{\text{argmin}} \ell(T)$ .

**end if**

Output  $T_{\text{apx}}$  and halt.

---

where the first equality follows from Lemmas 4, 5, and 6, and the last inequality follows from Lemma 6, and  $\ell(S) \leq \text{opt}(G, \ell)$ , and  $2\ell(T) \leq \text{opt}(G, \ell)$ . Thus  $T_{\text{apx}}$  is a 3/4-differential approximate tour. Note that minimum weighted 1- and 2-factors can be computed in polynomial time, and  $A_i$  and  $B_i$  ( $i = 1, 2$ ) can be computed in polynomial time. Thus **Algorithm TourEven** is polynomial, which completes the proof. □

## 4 Approximation for Odd Instances

In this section, we construct an approximation algorithm for TSP with an odd number of vertices. Our algorithm is much more involved than that in the even case. It first guesses a path  $P$  with three edges in an optimal tour, constructs eight path covers based on  $P$ , and extend each path cover to a tour in such a way that at least one of the eight tours guarantees 3/4-differential approximation ratio.

More precisely, for each path  $P$  with three edges, say,  $P = \{(v_1, v_2), (v_2, v_3), (v_3, v_4)\}$  with all  $v_i$ 's distinct, let  $S$  be a minimum weighted 2-factor among those containing  $P$ , let  $T$  be a minimum weighted path cover among those satisfying  $(v_1, v_2), (v_2, v_3) \in T$  and  $V_1(T) = V \setminus \{v_2\}$ , and let  $T'$  be a minimum weighted path cover among those satisfying  $(v_2, v_3), (v_3, v_4) \in T'$  and  $V_1(T') = V \setminus \{v_3\}$ . Assume that  $S$  is not a tour, i.e., it contains at least two cycles, since otherwise, is optimal, and hence ensures 3/4-differential approximability if some optimal tour contains  $P$ . We note that  $(S, T)$  and  $(S, T')$  are both valid pairs of spanning 2-matchings. We apply **Procedure FourPathCovers** to them, but not arbitrarily. Let us specify two cycles  $C^*$  and  $C^{**}$  in  $S$  such that  $P \subseteq C^*$  and  $P \cap C^{**} = \emptyset$ . We define two vertices  $v_0$  and  $v_5$  in  $V(C^*)$  such that  $v_0 \neq v_2$ ,  $v_5 \neq v_3$ , and  $(v_0, v_1), (v_4, v_5) \in C^*$ . By definition  $v_0 = v_4$  and  $v_5 = v_1$  hold if  $|C^*| = 4$ .



Furthermore, we define two edges  $f$  and  $f'$  in  $C^{**}$  that satisfy the properties in the next lemma.

Due to the space constraints, we omit the proofs of lemmas and constructions of tours from the results of **Procedure FourPathCovers**.

**Lemma 8.** *Let  $C^{**}$ ,  $T$  and  $T'$  be defined as above. Then there exist two edges  $f \in C^{**} \setminus T$  and  $f' \in C^{**} \setminus T'$  such that*

- (i) *they have a common endpoint  $q$ , and*
- (ii)  *$T \cup \{f\}$  and  $T' \cup \{f'\}$  are path covers.*

We note that  $f$  and  $f'$  in Lemma 8 might be identical, and (ii) in Lemma 8 implies that two pairs  $(S \setminus \{f\}, T \cup \{f\})$  and  $(S \setminus \{f'\}, T' \cup \{f'\})$  are valid.

Our algorithm uses **Procedure FourPathCovers** for  $(S, T)$  defined as above in such a way that edge  $e_1 = f$  is chosen in the first round and two edges  $e_1 = (v_3, v_4)$  and  $e_2 = (v_0, v_1)$  are chosen in the last round. Similarly, our algorithm uses **Procedure FourPathCovers** for  $(S, T')$  defined as above in such a way that edge  $e_1 = f'$  is chosen in the first round and two edges  $e_1 = (v_1, v_2)$  and  $e_2 = (v_4, v_5)$  are chosen in the last round. Let  $S_1, T_1, S_2$ , and  $T_2$  be four path covers obtained by **Procedure FourPathCovers** $(S, T)$ , and let  $S'_1, T'_1, S'_2$ , and  $T'_2$  be four path covers returned by **Procedure FourPathCovers** $(S, T')$ .

**Lemma 9.** *Let  $S, T, S_i$ , and  $T_i$  ( $i = 1, 2$ ) be defined as above. Then  $S_1, S_2, T_1$ , and  $T_2$  are path covers such that*

$$(i) \quad S_i \cup T_i = S \cup T \text{ and } S_i \cap T_i = S \cap T \text{ for } i = 1, 2, \tag{18}$$

$$(ii) \quad V_1(S_i) \text{ and } V_1(T_i) \text{ is a partition of } V \setminus \{v_2\} \text{ for } i = 1, 2, \tag{19}$$

$$(iii) \quad T_1 \setminus T_2 = \{(v_3, v_4)\}, T_2 \setminus T_1 = \{(v_0, v_1)\}, \\ \text{and } \{(v_1, v_2), (v_2, v_3)\} \in \mathcal{P}(T_1 \cap T_2), \text{ and} \tag{20}$$

$$(iv) \quad q \in V_1(S_1) \cap V_1(S_2), \tag{21}$$

where  $v_i \in V(C^*)$  ( $i = 0, \dots, 4$ ) are defined as above and  $q$  is a common endpoint of  $f$  and  $f'$  in Lemma 8.

Similarly, we have the following lemma.

**Lemma 10.** *Let  $S, T', S'_i$ , and  $T'_i$  ( $i = 1, 2$ ) be defined as above. Then  $S'_1, S'_2, T'_1$ , and  $T'_2$  are path covers such that*

$$(i) \quad S'_i \cup T'_i = S \cup T' \text{ and } S'_i \cap T'_i = S \cap T' \text{ for } i = 1, 2, \tag{22}$$

$$(ii) \quad V_1(S'_i) \text{ and } V_1(T'_i) \text{ is a partition of } V \setminus \{v_3\} \text{ for } i = 1, 2, \tag{23}$$

$$(iii) \quad T'_1 \setminus T'_2 = \{(v_1, v_2)\}, T'_2 \setminus T'_1 = \{(v_4, v_5)\}, \\ \text{and } \{(v_2, v_3), (v_3, v_4)\} \in \mathcal{P}(T'_1 \cap T'_2), \text{ and} \tag{24}$$

$$(iv) \quad q \in V_1(S'_1) \cap V_1(S'_2), \tag{25}$$

where  $v_i \in V(C^*)$  ( $i = 1, \dots, 5$ ) are defined as above and  $q$  is a common endpoint of  $f$  and  $f'$  in Lemma 8.

Let us then show how to construct edge sets  $A_i, B_i, A'_i,$  and  $B'_i$  (for  $i = 1, 2$ ), such that

$$(i) \quad S_i \cup A_i \text{ is a tour (for } i = 1, 2), \tag{26}$$

$$(ii) \quad T_i \cup B_i \text{ is a tour (for } i = 1, 2), \tag{27}$$

$$(iii) \quad S'_i \cup A'_i \text{ is a tour (for } i = 1, 2), \tag{28}$$

$$(iv) \quad T'_i \cup B'_i \text{ is a tour (for } i = 1, 2), \text{ and} \tag{29}$$

$$(v) \quad \sum_{i=1,2} (\ell(A_i) + \ell(B_i) + \ell(A'_i) + \ell(B'_i)) \leq 2\text{wor}(G, \ell) - 2\ell(v_2, v_3), \tag{30}$$

where  $\text{wor}(G, \ell)$  denotes the length of a longest tour of  $(G, \ell)$ .

We are now ready to describe our approximation algorithm, called **TourOdd**.

---

**Algorithm. TourOdd**

---

**Input:** A complete graph  $G = (V, E)$  with odd  $|V|$ , and an edge length function  $\ell : E \rightarrow \mathbb{R}_+$ .

**Output:** A tour  $T_{\text{apx}}$  in  $G$ .

**if**  $n < 17$  **then**

Compute an optimal tour  $T_{\text{opt}}$  of  $(G, \ell)$  by exhaustive search.

Output  $T_{\text{opt}}$  and halt.

**else**

$\mathcal{T} := \emptyset$ .

**for** each path  $P = \{(v_1, v_2), (v_2, v_3), (v_3, v_4)\}$  of length 3 in  $G$  **do**

Compute a minimum weighted 2-factor  $S$  among those containing  $P$ .

Compute a minimum weighted path cover  $T$  among those satisfying

$$(v_1, v_2), (v_2, v_3) \in T \text{ and } V_1(T) = V \setminus \{v_2\}.$$

Compute a minimum weighted path cover  $T'$  among those satisfying

$$(v_2, v_3), (v_3, v_4) \in T' \text{ and } V_1(T') = V \setminus \{v_3\}.$$

**if**  $S$  is a tour **then**

$$\mathcal{T} := \mathcal{T} \cup \{S\}.$$

**else**

$$S_1, T_1, S_2, T_2 := \text{FourPathCovers}(S, T).$$

$$S'_1, T'_1, S'_2, T'_2 := \text{FourPathCovers}(S, T').$$

Compute edge sets  $A_1, A_2, B_1, B_2, A'_1, A'_2, B'_1,$  and  $B'_2$ ,

which satisfy (26), (27), (28), (29), and (30).

$$\begin{aligned} \mathcal{T} := \mathcal{T} \cup \{ & S_1 \cup A_1, S_2 \cup A_2, T_1 \cup B_1, T_2 \cup B_2 \} \\ & \cup \{ S'_1 \cup A'_1, S'_2 \cup A'_2, T'_1 \cup B'_1, T'_2 \cup B'_2 \}. \end{aligned}$$

**end if**

**end for**

$$T_{\text{apx}} := \underset{T \in \mathcal{T}}{\text{argmin}} \ell(T).$$

Output  $T_{\text{apx}}$  and halt.

**end if**

---

Before analyzing the output of **Algorithm TourOdd**, let us evaluate  $\ell(S)$ ,  $\ell(T)$  and  $\ell(T')$ .

**Lemma 11.** *For a path  $P = \{(v_1, v_2), (v_2, v_3), (v_3, v_4)\}$ , let  $S$ ,  $T$  and  $T'$  be defined as above. If there exists an optimal tour that contains  $P$ , then*

$$2\ell(S) + \ell(T) + \ell(T') \leq 3\text{opt}(G, \ell) + \ell(v_2, v_3).$$

Let us describe our main result.

**Theorem 12.** *For a complete graph  $G = (V, E)$  with an odd number of vertices and an edge length function  $\ell : E \rightarrow \mathbb{R}_+$ , **Algorithm TourOdd** computes a 3/4-differential approximate tour of  $(G, \ell)$  in polynomial time.*

**Acknowledgments.** This work was partially supported by the joint project of Kyoto University and Toyota Motor Corporation, titled “Advanced Mathematical Science for Mobility Society” and by JSPS KAKENHI Grant Numbers JP19K22841, JP20H00609, and JP20H05967.

## References

1. Amano, Y., Makino, K.: A 3/4 differential approximation algorithm for traveling salesman problem (2020)
2. Arora, S.: Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *J. ACM* **45**(5), 753–782 (1998)
3. Ausiello, G., Bazgan, C., Demange, M., Paschos, V.T.: Completeness in differential approximation classes. *Int. J. Found. Comput. Sci.* **16**(06), 1267–1295 (2005)
4. Ausiello, G., Paschos, V.T.: Differential ratio approximation. In: Gonzalez, T.F. (ed.) *Handbook of Approximation Algorithms and Metaheuristics: Methodologies and Traditional Applications*, Volume 1, chap. 15, pp. 259–274. Chapman and Hall/CRC Computer and Information Science, Chapman and Hall/CRC (2018)
5. Bellman, R.: Dynamic programming treatment of the travelling salesman problem. *J. ACM (JACM)* **9**(1), 61–63 (1962)
6. Bland, R.G., Shallcross, D.F.: Large travelling salesman problems arising from experiments in X-ray crystallography: a preliminary report on computation. *Oper. Res. Lett.* **8**(3), 125–128 (1989)
7. Chlebík, Miroslav, Chlebíková, Janka: Approximation hardness of travelling salesman via weighted amplifiers. In: Du, Ding-Zhu., Duan, Zhenhua, Tian, Cong (eds.) *COCOON 2019*. LNCS, vol. 11653, pp. 115–127. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-26176-4\\_10](https://doi.org/10.1007/978-3-030-26176-4_10)
8. Cook, W.J.: *In Pursuit of the Traveling Salesman*. Princeton University Press, Princeton (2011)
9. Demange, M., Paschos, V.T.: On an approximation measure founded on the links between optimization and polynomial approximation theory. *Theoret. Comput. Sci.* **158**(1), 117–141 (1996)
10. Engebretsen, Lars, Holmerin, Jonas: Clique is hard to approximate within  $n^{1-o(1)}$ . In: Montanari, Ugo, Rolim, José D. P., Welzl, Emo (eds.) *ICALP 2000*. LNCS, vol. 1853, pp. 2–12. Springer, Heidelberg (2000). [https://doi.org/10.1007/3-540-45022-X\\_2](https://doi.org/10.1007/3-540-45022-X_2)

11. Escoffier, B., Monnot, J.: A better differential approximation ratio for symmetric TSP. *Theoret. Comput. Sci.* **396**(1), 63–70 (2008)
12. Grötschel, M., Jünger, M., Reinelt, G.: Optimal control of plotting and drilling machines: a case study. *Z. Oper. Res.* **35**(1), 61–84 (1991)
13. Hassin, R., Khuller, S.:  $z$ -approximations. *J. Algorithms* **41**(2), 429–442 (2001)
14. Held, M., Karp, R.M.: A dynamic programming approach to sequencing problems. *J. Soc. Ind. Appl. Math.* **10**(1), 196–210 (1962)
15. Karlin, A.R., Klein, N., Gharan, S.O.: A (slightly) improved approximation algorithm for metric TSP. In: *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pp. 32–45. STOC 2021, Association for Computing Machinery, New York, NY, USA (2021)
16. Lin, S., Kernighan, B.W.: An effective heuristic algorithm for the traveling-salesman problem. *Oper. Res.* **21**(2), 498–516 (1973)
17. Little, J.D.C., Murty, K.G., Sweeney, D.W., Karel, C.: An algorithm for the traveling salesman problem. *Oper. Res.* **11**(6), 972–989 (1963)
18. Mitchell, J.S.B.: Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP,  $k$ -MST, and related problems. *SIAM J. Comput.* **28**(4), 1298–1309 (1999)
19. Monnot, J.: Differential approximation results for the traveling salesman and related problems. *Inf. Process. Lett.* **82**(5), 229–235 (2002)
20. Monnot, J., Paschos, V.T., Toulouse, S.: Approximation algorithms for the traveling salesman problem. *Math. Methods Oper. Res.* **56**(3), 387–405 (2003)
21. Monnot, J., Paschos, V.T., Toulouse, S.: Differential approximation results for the traveling salesman problem with distances 1 and 2. *Eur. J. Oper. Res.* **145**(3), 557–568 (2003)
22. Monnot, J., Toulouse, S.: *The Traveling Salesman Problem and its Variations*, chap. 7, pp. 173–214. Wiley (2014)
23. Papadimitriou, C.H., Steiglitz, K.: *Combinatorial optimization: algorithms and complexity*. Courier Corporation (1998)
24. Punnen, A.P.: *The Traveling Salesman Problem: Applications, Formulations and Variations*, pp. 1–28. Springer, Boston (2007). [https://doi.org/10.1007/0-306-48213-4\\_1](https://doi.org/10.1007/0-306-48213-4_1)
25. Schrijver, A., et al.: *Combinatorial Optimization: Polyhedra and Efficiency*, vol. 24. Springer, Heidelberg (2003)
26. Shmoys, D.B., Lenstra, J.K., Kan, A.H.G.R., Lawler, E.L.: *The Traveling Salesman Problem*, A Wiley-Interscience publication, vol. 12. Wiley, New York (1985)



# Exact and Parameterized Algorithms for Restricted Subset Feedback Vertex Set in Chordal Graphs

Tian Bai<sup>(✉)</sup> and Mingyu Xiao

School of Computer Science and Engineering, University of Electronic Science  
and Technology of China, Chengdu, People's Republic of China  
tian.bai.cs@outlook.com, myxiao@uestc.edu.cn

**Abstract.** The RESTRICTED SUBSET FEEDBACK VERTEX SET problem (R-SFVS) takes a graph  $G = (V, E)$ , a terminal set  $T \subseteq V$ , and an integer  $k$  as the input. The task is to determine whether there exists a subset  $S \subseteq V \setminus T$  of at most  $k$  vertices, after deleting which no terminal in  $T$  is contained in a cycle in the remaining graph. R-SFVS is NP-complete even when the input graph is restricted to chordal graphs. In this paper, we show that R-SFVS in chordal graphs can be solved in time  $\mathcal{O}(1.1550^{|V|})$ , significantly improving all the previous results. As a by-product, we prove that the MAXIMUM INDEPENDENT SET problem parameterized by the edge clique cover number is fixed-parameter tractable. Furthermore, by using a simple reduction from R-SFVS to VERTEX COVER, we obtain a  $1.2738^k |V|^{\mathcal{O}(1)}$ -time parameterized algorithm and an  $\mathcal{O}(k^2)$ -kernel for R-SFVS in chordal graphs.

**Keywords:** Subset feedback vertex set · Chordal graphs · Exact exponential algorithms · Parameterized algorithms

## 1 Introduction

The FEEDBACK VERTEX SET problem (FVS), to delete at most  $k$  vertices from an  $n$ -vertex graph such that the remaining graph has no cycle, is one of the most extensively studied problems. The SUBSET FEEDBACK VERTEX SET problem (SFVS), firstly introduced by Even et al. [7], has also become a classic NP-complete problem. In this problem, we are further given a vertex subset  $T$  of the graph called *terminal set*, and we are asked to delete at most  $k$  vertices from the graph such that no terminal in  $T$  is contained in a cycle in the remaining graph. According to whether the terminal vertices in  $T$  can be deleted or not, there are two versions of SFVS: the restricted version, called R-SFVS, where terminal vertices in  $T$  are not allowed to be deleted, and the unrestricted version, simply denoted by SFVS, where any vertex in the graph can be deleted.

This work was supported by the National Natural Science Foundation of China (Grant No. 61972070).

FVS, SFVS, R-SFVS are closely related to several other important problems. For example, R-SFVS with  $|T| = 1$  generalizes the classical NODE MULTIWAY CUT problem [9].

The first non-trivial exact algorithm for SFVS dates back to the enumeration algorithm of Fomin et al. in 2014 [9]. They showed that the weighted version of SFVS can be solved in time  $\mathcal{O}(1.8638^n)$ . This suggests that the algorithm can also solve R-SFVS in the same time. Iwata et al. [17, 18] gave a single-exponential parameterized algorithm for SFVS running in time  $4^k n^{\mathcal{O}(1)}$ , which implies that this problem can be solved in  $\mathcal{O}(1.75^n)$  using the techniques by Fomin et al. [8]. Recently, Hols and Kratsch showed that SFVS has a randomized polynomial kernelization with  $\mathcal{O}(k^9)$  vertices [14]. In comparison, FVS can be solved in time  $3.460^k n^{\mathcal{O}(1)}$  [16], which indicates that FVS can be solved in time  $\mathcal{O}(1.711^n)$  [8]. Besides, FVS admits a quadratic kernel [15, 24], whereas whether there is a polynomial kernel for SFVS is still unknown.

FVS and SFVS have also been studied in several graph classes, such as chordal graphs and split graphs. Chordal graphs are the graphs without containing induced cycles of length four or larger, and split graphs are a subclass of the chordal graphs whose vertex set can be partitioned into a clique and an independent set. Both R-SFVS and SFVS remain NP-complete even in split graphs [9], while FVS in chordal graphs are polynomial-time solvable [28]. The best-known algorithm for the weighted version of SFVS in chordal graphs runs in time  $\mathcal{O}(1.6708^n)$  [12]. Later, Chitnis et al. [3] gave an algorithm for SFVS in chordal graphs running in time  $\mathcal{O}(1.6181^n)$ . Since SFVS in chordal graphs is a special case of 3-HITTING SET, we know that it can be solved in time  $\mathcal{O}(1.5182^n)$  [8] and  $2.076^k n^{\mathcal{O}(1)}$  [25]. Very recently, Philip et al. [21] broke the bounds for 3-HITTING SET by giving an  $\mathcal{O}(1.5^n)$ -time exact algorithm and a  $2^k n^{\mathcal{O}(1)}$ -time parameterized algorithm for SFVS in chordal graphs. They also gave a quadratic kernel for SFVS in split graphs. Since most above algorithms are enumeration algorithms and branching-and-searching algorithms, they can be easily extended to deal with the restricted version R-SFVS. In this paper, we will focus on R-SFVS in chordal graphs and try to make further improvements on this problem.

The main technical contributions of this paper are two-fold. Firstly, we present a parameter-preserved polynomial-time reduction from VERTEX COVER to R-SFVS in chordal graphs. This reduction directly derives an  $\mathcal{O}(1.1996^{n-|T|})$ -time algorithm and an  $\mathcal{O}^*(1.2738^k)$ -time<sup>1</sup> algorithm for R-SFVS in chordal graphs by using the previous results for VERTEX COVER [2, 4, 20, 27]. Based on this reduction, we also obtain a kernel for  $\mathcal{O}(k^2)$  vertices for R-SFVS. Second, we further improve the running time bound of the exact algorithms by using more techniques. Specifically, we prove that MAXIMUM INDEPENDENT SET can be solved in time  $\mathcal{O}^*(2^\tau)$  given an edge clique cover of size  $\tau$ . With the help of this algorithm, we show that R-SFVS in chordal graphs can be solved in time  $\mathcal{O}^*(2^{|T|})$ . By doing a tradeoff between the two algorithms with running time

<sup>1</sup> The  $\mathcal{O}^*(\cdot)$  notation hides all polynomial factors, i.e.,  $f(n) = \mathcal{O}^*(g(n))$  means  $f(n) = g(n)n^{\mathcal{O}(1)}$ .

bounds  $\mathcal{O}(1.1996^{n-|T|})$  and  $\mathcal{O}^*(2^{|T|})$ , finally, we demonstrate that R-SFVS in chordal graphs can be solved in time  $\mathcal{O}(1.1550^n)$ . Due to page limitations, the proofs of some lemmas and theorems marked with ‘ $\blacklozenge$ ’ may be omitted. The full proofs can be found in the full version of this paper.

## 2 Preliminaries

### 2.1 Graphs

We work with an undirected graph  $G = (V, E)$  without parallel edges, where  $|V| = n$  and  $|E| = m$ . Let  $X \subseteq V$  be a subset of vertices. The *neighbor set* of  $X$  is denoted by  $N(X) := \{v \in V \setminus X : \exists x \in X \text{ s.t. } (v, x) \in E\}$ , and the *closed neighbor set* of  $X$  is  $N[X] := N(X) \cup X$ . The subgraph of  $G$  induced by  $X$  is denoted by  $G[X]$ . The subgraph obtained from  $G$  by removing  $X$  together with edges incident on any vertex in  $X$  is denoted by  $G - X$ . For ease of notation, we may denote a singleton set  $\{v\}$  by  $v$ . The *degree* of a vertex  $v$  in  $G$  is defined as  $\deg(v) := |N(v)|$ . An edge  $e$  is a *bridge* if it is not contained in any cycle of  $G$ . A *separator* of a graph is a vertex set such that the deletion of it will increase the number of connected components of the graph. The shorthand  $[r]$  is expressed as the set  $\{1, 2, \dots, r\}$  for  $r \in \mathbb{N}^+$ .

For an undirected graph  $G = (V, E)$ , a subset  $Q \subseteq V$  of vertices of the graph  $G$  is a *clique* if every pair of distinct vertices in  $Q$  is connected by a unique edge. A vertex  $v$  is *simplicial* in  $G$  if  $N[v]$  is a clique [5]. A maximal clique in  $G$  is a *simplicial clique* if it contains a simplicial vertex.

An *edge clique cover* of a graph  $G$  is a set of cliques that covers all the edges in  $G$ . The *edge clique cover number*, a.k.a. the *intersection number*, is the smallest number of such cliques covering all edges in the graph.

### 2.2 Chordal Graphs and Split Graphs

A *chord* of a cycle is an edge between two non-consecutive vertices of the cycle. A graph  $G$  is *chordal* if every cycle of length at least 4 has a chord. The following properties of chordal graphs will be used in the paper. For a chordal graph  $G$ , the following facts hold [5]:

1. Every induced subgraph of  $G$  is chordal.
2. Every minimal separator of  $G$  is a clique.

For a connected chordal graph  $G$ , let  $\mathcal{Q}_G$  be the set of all maximal cliques in  $G$ . A *clique graph* of  $G$  is an edge-weighted undirected graph  $(\mathcal{Q}_G, \mathcal{E}_G, \sigma)$  with  $\sigma: \mathcal{E}_G \rightarrow \mathbb{N}$  satisfying that an edge  $Q_1Q_2 \in \mathcal{E}_G$  if  $Q_1 \cap Q_2$  is a minimal separator and  $\sigma(Q_1Q_2) := |Q_1 \cap Q_2|$ . A *clique tree*  $\mathcal{T}_G$  of  $G$  is a maximum spanning tree of the clique graph of  $G$  and the following facts hold [1, 11, 26]:

1. Each leaf node of a clique tree  $\mathcal{T}_G$  is a simplicial clique in  $G$ .
2. For each edge  $Q_1Q_2$  in  $\mathcal{T}_G$ ,  $Q_1 \cap Q_2$  separates the graph  $G$ .

Whether a graph  $G$  is chordal or not can be checked in time  $\mathcal{O}(n + m)$  [22]. If  $G$  is chordal, the number of maximal cliques in  $G$  is at most  $n$  [10], and all of the maximal cliques can be listed in time  $\mathcal{O}(n + m)$  [11]. These properties will be used in our algorithm.

A graph is a *split graph* if its vertex set can be partitioned into two parts, one of which induces a clique and the other of which induces an independent set [23]. Such a partition, is called a *split partition*. Every split graph is chordal, and whether a graph  $G$  is a split graph or not can also be easily checked in  $\mathcal{O}(n + m)$  time by the definition.

### 2.3 Subset Feedback Vertex Set in Chordal Graphs

Given a terminal set  $T \subseteq V$  of the graph  $G$ . A cycle in  $G$  is a  $T$ -cycle if it contains a terminal in  $T$ . A graph is a  $T$ -forest if there is no  $T$ -cycle. A *subset feedback vertex set* of a graph with a terminal set  $T$  is a subset of  $V \setminus T$  whose deletion makes the remaining graph a  $T$ -forest. In this paper, we mainly study R-SFVS when the input is restricted to chordal graphs. The problem is formally defined as follows.

R-SFVS IN CHORDAL GRAPHS  
**Input:** A chordal graph  $G = (V, E)$ , a terminal set  $T \subseteq V$ , and an integer  $k$ .  
**Output:** Determine whether there is a subset feedback vertex set  $S \subseteq V \setminus T$  of size at most  $k$  vertices.

When the input graph is restricted to split graphs, we call the problem R-SFVS IN SPLIT GRAPHS.

A  $T$ -triangle is a  $T$ -cycle of length three. The following lemma states that we can change the subset feedback vertex set problem to the problem of finding a subset of vertices hitting all  $T$ -triangles instead of all  $T$ -cycles.

**Lemma 1** ([21]). *Let  $G$  be a chordal graph and  $T \subseteq V$  be the terminal set. A vertex set  $S \subseteq V$  is a subset feedback vertex set of  $G$  if and only if  $G - S$  contains no  $T$ -triangles.*

### 2.4 A Technique for Algorithm Design

Our fast algorithm for R-SFVS IN CHORDAL GRAPHS adopts the following framework, which makes the tradeoff between two known parameterized algorithms. Notably, this framework can be used to design algorithms for more problems.

**Lemma 2.** (♦) *Let  $P$  be a problem, and  $n, \ell$  be two parameters of the input instance of  $P$  such that  $n \geq \ell$ . If  $P$  can be solved in time  $\mathcal{O}^*(\alpha^\ell)$  and  $\mathcal{O}^*(\beta^{n-\ell})$  respectively for some constants  $\alpha, \beta > 1$ , then the problem  $P$  can be solved in time  $\mathcal{O}^*(2^{\gamma n})$ , where  $\gamma = \frac{\log \alpha \log \beta}{\log \alpha \beta}$ .*



We will use Lemma 2 to design a fast exact algorithm for R-SFVS IN CHORDAL GRAPHS. In particular, we design two parameterized algorithms with the parameter being  $n - |T|$  and  $n$  in Sects. 3 and 4, respectively. Then we obtain an exact algorithm for it by doing a tradeoff between these two parameterized algorithms with Lemma 2.

### 3 Reductions Between R-SFVS IN CHORDAL GRAPHS and VERTEX COVER

In this section, we show some relations between R-SFVS IN CHORDAL GRAPHS and VERTEX COVER problem. By using known results for VERTEX COVER, we may quickly get some results for R-SFVS IN CHORDAL GRAPHS. VERTEX COVER is one of Karp's 21 NP-complete problems [19]. In VERTEX COVER, we are given a graph  $G = (V, E)$  and a positive integer  $k$ , and the object is to determine whether there exists a set  $S \subseteq V$  of size at most  $k$  such that  $G - S$  is an independent set. We have the following reduction.

**Lemma 3.** (♦) *Any instance  $\mathcal{I} = (G = (V, E), T, k)$  of R-SFVS IN CHORDAL GRAPHS can be polynomially reduced to an instance  $\mathcal{I}' = (G' = (V', E'), k')$  of VERTEX COVER such that  $\mathcal{I}$  is a YES-instance if and only if  $\mathcal{I}'$  is a YES-instance, where  $|V'| \leq |V| - |T|$  and  $k' \leq k$ .*

Since VERTEX COVER can be solved in time  $\mathcal{O}^*(1.2738^k)$  [2] and in time  $\mathcal{O}^*(1.19951^n)$  [27], respectively, we derive the following corollary.

**Corollary 1.** R-SFVS IN CHORDAL GRAPHS can be solved in time  $\mathcal{O}^*(1.2738^k)$  and  $\mathcal{O}^*(1.19951^{n-|T|})$ , respectively.

The second result in Corollary 1, together with Lemma 2, will be used to obtain a fast exact algorithm for R-SFVS IN CHORDAL GRAPHS later.

We have given a reduction from R-SFVS IN CHORDAL GRAPHS to VERTEX COVER in Corollary 1. There is also a simple reduction from VERTEX COVER to R-SFVS IN SPLIT GRAPHS introduced by Fomin et al. [9]. Here we slightly refine the instance size of R-SFVS IN SPLIT GRAPHS in the reduction.

**Lemma 4.** (♦) *Any instance  $\mathcal{I} = (G = (V, E), k)$  of VERTEX COVER can be polynomially reduced to an instance  $\mathcal{I}' = (G' = (V', E'), T', k')$  of R-SFVS IN SPLIT GRAPHS such that  $\mathcal{I}$  is a YES-instance if and only if  $\mathcal{I}'$  is a YES-instance, where  $|V'| \leq |V| + |V|^2/4$ ,  $|E'| \leq 3|E|$  and  $k' \leq k$ .*

Recall that split graphs belong to chordal graphs. The two reductions in Lemmas 3 and 4 preserve the parameter  $k$ , which shows the equivalence between VERTEX COVER and R-SFVS IN CHORDAL GRAPHS (or R-SFVS IN SPLIT GRAPHS) in parameterized complexity with parameter  $k$ .

**Corollary 2.** R-SFVS IN CHORDAL GRAPHS (or R-SFVS IN SPLIT GRAPHS) can be solved in  $f(k)n^{\mathcal{O}(1)}$  time if and only if VERTEX COVER can be solved in  $f(k)n^{\mathcal{O}(1)}$  time.

The two reductions in Lemmas 3 and 4 can also be used to obtain kernels for R-SFVS IN CHORDAL GRAPHS and R-SFVS IN SPLIT GRAPHS.

**Theorem 1.** R-SFVS IN CHORDAL GRAPHS and R-SFVS IN SPLIT GRAPHS admit a kernel with at most  $k^2 + 2k$  vertices and  $6k^2$  edges.

*Proof.* Let  $\mathcal{I} = (G = (V, E), T, k)$  be an instance of R-SFVS IN CHORDAL GRAPHS or R-SFVS IN SPLIT GRAPHS. First, we reduce  $\mathcal{I}$  to an equivalent instance  $\mathcal{I}_0 = (G_0 = (V_0, E_0), k_0)$  of VERTEX COVER by Lemma 3. It holds that  $k_0 \leq k$ . Second, we apply known kernelization algorithms for VERTEX COVER [4, 20] on  $\mathcal{I}_0$  to get a  $2k$ -kernel:  $\mathcal{I}_1 = (G_1 = (V_1, E_1), k_1)$  satisfying that  $k_1 \leq k_0 \leq k$ , and  $|V_1| \leq 2k_1$ . Third, we apply the reduction in Lemma 4 to transfer  $\mathcal{I}_1$  to an equivalent instance  $\mathcal{I}_2 = (G_2 = (V_2, E_2), T_2, k_2)$  of R-SFVS IN SPLIT GRAPHS, which is the desired. We have that  $k_2 \leq k_1 \leq k_0 \leq k$ ,  $|V_2| \leq |V_1| + |V_1|^2/4 \leq 2k_1 + (2k_1)^2/4 \leq k^2 + 2k$ , and  $|E_2| \leq 3|E_1| \leq 3|V_1|^2/2 \leq 3(2k_1)^2/2 \leq 6k^2$ .

Therefore, we get a kernel with the claimed size bound. □

Generally speaking, a kernel for R-SFVS IN CHORDAL GRAPHS does not imply the kernel of the same size for R-SFVS IN SPLIT GRAPHS, and the opposite is also not true.

## 4 A Fast Exact Algorithm for R-SFVS IN CHORDAL GRAPHS

Our idea is to use Lemma 2 to design a fast exact algorithm for R-SFVS IN CHORDAL GRAPHS. We already have an  $\mathcal{O}^*(1.19951^{n-|T|})$ -time algorithm for R-SFVS IN CHORDAL GRAPHS. Next, we are going to design an  $\mathcal{O}^*(2^{|T|})$ -time algorithm for this problem. This part is more technically involved.

Our algorithm needs to solve the maximum independent set problem based on a given edge clique cover of the input graph. We first introduce this algorithm.

### 4.1 Max Independent Set Based on Edge Clique Cover

We now give an algorithm finding a maximum independent set.

**Lemma 5.** Given an edge clique cover  $\mathcal{C}$  of a graph  $G = (V, E)$ , a maximum independent set of  $G$  can be computed in time  $\mathcal{O}^*(2^{|\mathcal{C}|})$ .

*Proof.* Suppose that  $\mathcal{C} = \{Q_1, Q_2, \dots, Q_\tau\}$  is an edge clique cover of  $G$  of size  $\tau$ . We define a vertex-label function  $\sigma: v \mapsto (\sigma_1(v), \sigma_2(v), \dots, \sigma_\tau(v))$  satisfying

$$\sigma_i(v) = \begin{cases} 0, & v \in Q_i \\ 1, & v \notin Q_i. \end{cases} \tag{1}$$

For a subset  $V' \subseteq V$ , we also define

$$\sigma(V') := \sum_{v \in V'} \sigma(v) = \left( \sum_{v \in V'} \sigma_1(v), \sum_{v \in V'} \sigma_2(v), \dots, \sum_{v \in V'} \sigma_\tau(v) \right). \quad (2)$$

For two vectors  $\mathbf{x} = (x_1, x_2, \dots, x_\tau)$  and  $\mathbf{y} = (y_1, y_2, \dots, y_\tau)$ , we write  $\mathbf{x} \leq \mathbf{y}$  if it holds that  $x_i \leq y_i$  for each  $i \in [\tau]$ . We will also write  $\mathbf{a} \not\leq \mathbf{b}$  to denote that  $\mathbf{a} \leq \mathbf{b}$  does not hold. We use  $\mathbf{0}$  (resp.,  $\mathbf{1}$ ) to denote the all-zeros vector (resp., all-ones vector).

Let  $v$  and  $u$  be a pair of vertices in the graph  $G$ . If there is an edge between them, then this edge should be in at least one clique in  $\mathcal{C}$  since  $\mathcal{C}$  is an edge clique cover. Thus the two vertices  $v$  and  $u$  should appear in at least one of the same clique in  $\mathcal{C}$ . It holds that  $\sigma_i(\{v, u\}) \geq 2$  for at least one  $i \in [\tau]$ . On the flip hand, if there is an index  $i \in [\tau]$  such that  $\sigma_i(\{v, u\}) \geq 2$ , then  $v$  and  $u$  are in the same clique  $Q_i$  and thus there is an edge between  $v$  and  $u$ . Then we conclude that a vertex subset  $V' \subseteq V$  is an independent set if and only if  $\sigma(V') \leq \mathbf{1}$ .

Next, we change our maximum independent set problem to the problem of finding a maximum set  $V' \subseteq V$  such that  $\sigma(V') \leq \mathbf{1}$ .

We use a dynamic programming algorithm to solve this problem. Fix an order of vertices  $V = \{v_1, v_2, \dots, v_n\}$ . Let  $V_0 := \emptyset$  and  $V_j := V_{j-1} \cup \{v_j\}$  for each  $j \in [n]$ . Let  $\mathcal{X}$  be the set of vectors  $\mathbf{x}$  such that the dimension of  $\mathbf{x}$  is  $\tau$  and  $\mathbf{x} \leq \mathbf{1}$ . We know that  $\mathcal{X}$  contains  $2^\tau$  elements.

For each  $i \in [n]$  and each  $\mathbf{x} \in \mathcal{X}$ , we use  $f[\mathbf{x}, i]$  to denote the size of the maximum set  $V' \subseteq V_i$  such that  $\sigma(V') = \mathbf{x}$ , and using the following recurrence relation to compute it:  $f[\mathbf{0}, 0] = 0$  and  $f[\mathbf{x} \neq \mathbf{0}, 0] = \infty$ , and for every  $i \geq 1$ ,

$$f[\mathbf{x}, i] = \begin{cases} \max \{ f[\mathbf{x} - \sigma(v_i), i - 1] + 1, f[\mathbf{x}, i - 1] \}, & \text{if } \sigma(v_i) \leq \mathbf{x} \\ f[\mathbf{x}, i - 1], & \text{if } \sigma(v_i) \not\leq \mathbf{x}. \end{cases} \quad (3)$$

Based on the bottom-up method, we compute  $f[\mathbf{x}, i]$  by using (3) when  $f[\mathbf{x}', i']$  has been computed for all  $\mathbf{x}' \in \mathcal{X}$  and all  $i' < i$ . For each fixed  $\mathbf{x}$  and  $i$ , it takes only constant time to compute  $f$  and store the result. There are  $n + 1$  possible values for  $i$  and  $2^\tau$  candidates for  $\mathbf{x}$ . So it takes  $\mathcal{O}(n2^\tau)$  time.

After computing  $f[\mathbf{x}, i]$ , we can solve our problem directly since the maximum size of  $V' \subseteq V$  such that  $\sigma(V') \leq \mathbf{1}$  is  $\max_{\mathbf{x} \in \mathcal{X}} f[\mathbf{x}, n]$ . □

Finding an edge clique cover of size  $\tau$  can be solved in time  $\mathcal{O}^*(2^{2^\tau})$  [13]. Hence, Lemma 5 also yields the following tractable result. Note that MAXIMUM INDEPENDENT SET parameterized by the solution size  $k$  is W[1]-complete [6].

**Corollary 3.** *MAXIMUM INDEPENDENT SET parameterized by the edge clique cover number is fixed-parameter tractable.*

## 4.2 Some Reduction Rules

We are ready to introduce our algorithm for R-SFVS IN CHORDAL GRAPHS, which will contain three parts. In the first part, we introduce some reduction rules to simplify the input graph. In the second part, we solve a special case of the problem with some restrictions on the input graph (including the class of split graphs). In the last part, we solve the problem in general chordal graphs through a dividing procedure. Now, we present our reduction rules. When introducing one reduction rule, we assume that all previous reduction rules can not be applied to the current instance.

**Reduction Rule 1.** *If there is a triangle in  $G$  with three vertices being terminals from  $T$ , report the instance as a NO-instance directly.*

**Reduction Rule 2.** *If there is a triangle in  $G$  with exactly two vertices being terminals from  $T$ , delete the third vertex of the triangle and decrease  $k$  by 1.*

Recall that a bridge is an edge not contained in any cycle. The remaining graph is still chordal after removing a bridge from a chordal graph. Since a bridge is not contained in any cycle, we have Reduction Rule 3.

**Reduction Rule 3.** *Remove all bridges in the graph.*

**Reduction Rule 4.** *Remove all vertices not contained in any  $T$ -triangle.*

**Reduction Rule 5.** *If there is a simplicial non-terminal  $v$  adjacent to exactly one terminal  $t \in T$ , then delete all vertices in  $N(v) \setminus \{t\}$  from the graph and decrease  $k$  by  $|N(v)| - 1$ .*

**Lemma 6.** (♦) *An instance  $(G, T, k)$  of R-SFVS IN CHORDAL GRAPHS is a YES-instance if and only if the instance  $(G', T, k')$  after applying Reduction Rule 5 is a YES-instance.*

**Definition 1 (Reduced Instances).** *An instance of R-SFVS IN CHORDAL GRAPHS is called reduced, if none of Reduction Rules 1 to 5 can be applied to it.*

**Lemma 7.** (♦) *For any input instance of R-SFVS IN CHORDAL GRAPHS, we can iteratively apply Reduction Rules 1 to 5 to transfer it to a reduced instance in time  $\mathcal{O}(n^3)$ .*

**Lemma 8.** (♦) *A reduced instance  $(G = (V, E), T, k)$  of R-SFVS IN CHORDAL GRAPHS holds the following properties:*

- (a)  $N[T] = V$ ;
- (b) each clique in the graph contains at most one terminal;
- (c) each simplicial clique contains only one simplicial vertex that is a terminal.

### 4.3 A Special Case

In this subsection, we design an  $\mathcal{O}^*(2^{|T|})$ -time algorithm for a special R-SFVS IN CHORDAL GRAPHS where each terminal is contained in only one maximal clique of the input graph. Note that a vertex is contained in exactly one maximal clique if and only if the vertex is simplicial. We denote the algorithm by **SubALG**, and it will be used as a sub-algorithm in our algorithm for R-SFVS IN CHORDAL GRAPHS.

The main idea of **SubALG** is as follows. We first apply the reduction rules to get a reduced instance  $(G = (V, E), T, k)$ . Notably, if the original graph satisfies that each terminal is in exactly one maximal clique, this property still holds for the reduced graph (obtained by deleting some vertices and bridges). Next, we solve the reduced instance by using Lemma 5.

According to the reduction in Lemma 3, we construct a graph  $G' = (V', E')$  based on the reduced instance  $(G, T, k)$  such that  $G$  has a subset feedback vertex set of size  $k$  if and only if graph  $G'$  has an independent set of size  $|V \setminus T| - k$ . Recall that the vertex set  $V' = \{v' : v \in V \setminus T\}$ . Besides, an edge  $u'v' \in E'$  if and only if there is an edge  $uv \in E$  together with a terminal forming a  $T$ -triangle. The following lemma shows the relation between  $G$  and  $G'$ .

**Lemma 9.** ( $\blacklozenge$ ) *Let  $(G, T, k)$  be a reduced instance. There is an edge clique cover  $\mathcal{C}$  of  $G'$  with size  $|\mathcal{C}| = |T|$ . In addition,  $I'$  is an independent set of  $G'$  if and only if  $V \setminus (T \cup I)$  is a subset feedback vertex set of  $G$ , where  $I = \{v : v' \in I'\}$ .*

By Lemma 9, to find a minimum subset feedback vertex set of  $G$ , we only need to find a maximum independent set in  $G'$ . We apply the algorithm in Lemma 5 on the graph  $G'$  with the edge clique cover  $\mathcal{C}$  to find a maximum independent set  $I'$  of  $G'$  in  $\mathcal{O}^*(2^{|\mathcal{C}|})$  time, where  $|\mathcal{C}| = |T|$ . Thus, we get a minimum subset feedback vertex set  $V \setminus (T \cup I)$  of  $G$  by adopting the algorithm **SubALG**.

**Lemma 10.** *For the special case of R-SFVS IN CHORDAL GRAPHS where each terminal is in exactly one maximal clique of the input graph, algorithm **SubALG** solves the problem in  $\mathcal{O}^*(2^{|T|})$  time.*

Lemma 10 can also be directed used to solve R-SFVS IN SPLIT GRAPHS within the same running time bound.

**Corollary 4.** ( $\blacklozenge$ ) *R-SFVS IN SPLIT GRAPHS can be solved in  $\mathcal{O}^*(2^{|T|})$  time.*

### 4.4 Subset Feedback Vertex Set in Chordal Graphs

For general chordal graphs, we can also solve the problem by using Lemma 5, exactly as what we do in algorithm **SubALG**, in  $\mathcal{O}^*(2^{|\mathcal{C}|})$  time. Similar to Lemma 9,  $|\mathcal{C}|$  is equal to the number of the maximal cliques containing terminals in  $G$ . However, for general chordal graphs, it is possible that  $|\mathcal{C}| > |T|$  since a terminal may be contained in more than one maximal clique in the graph. To get our expected running time bound  $\mathcal{O}^*(2^{|T|})$ , we need to deal with the case where some terminal is contained in several maximal cliques.

In our algorithm, we also first apply reduction rules on the input instance to get a reduced instance  $(G, T, k)$ . From now on, we assume that the graph is reduced. Then we construct a clique tree  $\mathcal{T}_G$  of  $G$  with the set of maximal cliques  $\mathcal{Q}_G$  in linear time by using the algorithm in [11]. A terminal in  $T$  is called an *inter terminal* if it belongs to at least two different cliques in  $\mathcal{Q}_G$ . A maximal clique in  $\mathcal{Q}_G$  is called an *inter clique* if it contains an inter terminal. An inter clique  $Q \in \mathcal{Q}_G$  is called a *dividing clique* if after deleting  $Q$  from the clique tree  $\mathcal{T}_G$ , there is at least one connected component containing no inter cliques in  $\mathcal{T}_G$ . Such connected component is called a *good branch* (with respect to  $Q$ ).

After computing the clique tree, the algorithm checks whether there is an inter clique. If there is no inter clique, we can directly solve the instance in  $\mathcal{O}^*(2^{|T|})$  time by Lemma 10; otherwise, we employ a divide-and-conquer technique to resolve it. We will split the graph at a dividing clique and ensure that a part of the graph can be solved by using Lemma 10.

Now, we assume that there exist some inter cliques. Then the dividing cliques also exist. Let  $Q$  be a dividing clique and  $B_Q \subseteq V$  be the union of vertices in all maximal cliques in a good branch w.r.t  $Q$ . Assume that  $Q = \{t, v_1, v_2, \dots, v_l\}$ , where  $t$  is the unique terminal in  $Q$  and  $v_i$  ( $i \in [l]$ ) are other vertices. We let  $X_0 := B_Q \setminus Q$  and  $X_i := X_0 \cup \{v_i\}$  for  $i \in [l]$ . Note that  $G[X_i]$  for each  $i \in [l]$  is an induced subgraph of  $G$  such that each terminal is in at most one maximal clique. We know that the  $T \cap X_i = T \cap X_0$  for every index  $i \in [l]$ . Thus, we can find a minimum subset feedback vertex of  $G[X_i]$  with terminal set  $T \cap X_i$  in  $\mathcal{O}^*(2^{|T \cap X_0|})$  time by Lemma 10. We also use  $s_i$  to denote the size of the minimum subset feedback vertex of  $G[X_i]$  with terminal set  $T \cap X_0$ .

Following structural properties are essential for our algorithm’s correctness.

**Lemma 11.** (♦) *For each  $i \in [l]$ , it holds that  $s_0 \leq s_i \leq s_0 + 1$ .*

**Lemma 12.** (♦) *If  $s_i = s_0 + 1$  holds for some index  $i$ , then there is a minimum subset feedback vertex set of  $G$  containing vertex  $v_i$ .*

**Lemma 13.** (♦) *If  $s_i = s_0$  holds for each  $i \in [l]$ , then for any minimum subset feedback vertex set  $S$  of  $G$ , it holds that  $|S \cap X_0| = s_0$ .*

**Lemma 14.** (♦) *If  $s_i = s_0$  holds for each  $i \in [l]$ , then  $\mathcal{I} = (G, T, k)$  is a YES-instance if and only if  $\mathcal{I}_0 = (G - X_0, T \setminus X_0, k - s_0)$  is a YES-instance. Furthermore, if  $S'$  is a subset feedback vertex set of  $\mathcal{I}_0$  with size  $k - s_0$ , then  $S = S' \cup S''$  is a subset feedback vertex set of  $\mathcal{I}$  with size  $k$ , where  $S''$  is a minimum subset feedback vertex set of  $G[(X_0 \cup Q) \setminus (S' \cup \{t\})]$  with the terminal set  $X_0 \cap T$ .*

Based on Lemmas 12 and 14, our algorithm will deal with dividing cliques in the following way. Let  $Q$  be a dividing clique in  $G$ .

**Dividing Procedure:** The algorithm will iteratively check whether  $s_i > s_0$  for  $i \in [l]$  by SubALG. If yes, remove  $v_i$  from the graph and decrease  $k$  by 1 according to Lemma 12. When  $s_i = s_0$  holds for each non-terminal vertex  $v_i$  in  $Q$ , we remove  $X_0$  and decrease  $k$  by  $s_0$  according to Lemma 14.

**Theorem 2.** R-SFVS IN CHORDAL GRAPHS *can be solved in time*  $\mathcal{O}^*(2^{|T|})$ .

*Proof.* The algorithm first computes a clique tree of the graph and checks whether there is an inter clique. If there is no inter clique, then the instance satisfies the condition in Lemma 10, and we employ the algorithm **SubALG** to resolve the instance in  $\mathcal{O}^*(2^{|T|})$  time. If there exists an inter clique, there is a dividing clique  $Q$ , and then we execute the dividing procedure. To do this, we need to compute the minimum subset feedback vertex set  $S_i$  of  $G[X_i]$  for each  $i \in [l]$ . Each sub-instance satisfies the condition in Lemma 10, and then each of them can be computed in  $\mathcal{O}^*(2^{|T \cap X_0|})$  by **SubALG**. So the dividing procedure takes  $2^{|T \cap X_0|} n^c$  time to transfer an instance with  $|T|$  terminals to an instance with  $|T \setminus X_0|$  terminals, where  $c$  is a constant large enough. Let  $R(|T|)$  denote the running time of our algorithm on instances with  $|T|$  terminals. It follows the recurrence relation

$$R(|T|) \leq 2^{|T \cap X_0|} n^c + R(|T| - |T \cap X_0|), \quad (4)$$

which yields the total running time  $R(|T|) = \mathcal{O}^*(2^{|T|})$ .  $\square$

Based on Lemma 2, Corollary 1, and Theorem 2, we get the following result. Let  $\alpha = 2$  and  $\beta = 1.19951$  in Lemma 2. Then  $2^\gamma = 2^{\log \alpha \log \beta / \log \alpha \beta} < 1.1550$ .

**Theorem 3.** R-SFVS IN CHORDAL GRAPHS *can be solved in time*  $\mathcal{O}(1.1550^n)$ .

## 5 Conclusion

According to whether the terminals are allowed to be deleted or not, we can define unrestricted and restricted versions SFVS and R-SFVS. Both of them remain NP-hard in split graphs. There are certain relations between them. In general graphs, it is easy to reduce one of them to the other one, even preserving the solution size. However, when the input graphs are restricted to chordal graphs, we do not find a parameter-preserved reduction from SFVS IN CHORDAL GRAPHS to R-SFVS IN CHORDAL GRAPHS. Intuitively, SFVS IN CHORDAL GRAPHS is a special case of 3-HITTING SET, whereas R-SFVS IN CHORDAL GRAPHS is a special case of 2-HITTING SET (i.e., VERTEX COVER). It also implies that SFVS IN CHORDAL GRAPHS might be harder than R-SFVS IN CHORDAL GRAPHS.

## References

1. Buneman, P.: A characterisation of rigid circuit graphs. *Discret. Math.* **9**(3), 205–212 (1974)
2. Chen, J., Kanj, I.A., Xia, G.: Improved upper bounds for vertex cover. *Theor. Comput. Sci.* **411**(40–42), 3736–3756 (2010)
3. Chitnis, R., Fomin, F.V., Lokshtanov, D., Misra, P., Ramanujan, M.S., Saurabh, S.: Faster exact algorithms for some terminal set problems. *J. Comput. Syst. Sci.* **88**, 195–207 (2017)

4. Dehne, F., Fellows, M., Rosamond, F., Shaw, P.: Greedy localization, iterative compression, and modeled crown reductions: new FPT techniques, an improved algorithm for SET SPLITTING, and a Novel  $2k$  Kernelization for VERTEX COVER. In: Downey, R., Fellows, M., Dehne, F. (eds.) IWPEC 2004. LNCS, vol. 3162, pp. 271–280. Springer, Heidelberg (2004). [https://doi.org/10.1007/978-3-540-28639-4\\_24](https://doi.org/10.1007/978-3-540-28639-4_24)
5. Dirac, G.A.: On rigid circuit graphs. *Abh. Math. Sem. Univ. Hamburg* **25**(1), 71–76 (1961)
6. Downey, R.G., Fellows, M.R.: Fixed-parameter tractability and completeness II: on completeness for  $W[1]$ . *Theor. Comput. Sci.* **141**(1&2), 109–131 (1995)
7. Even, G., Naor, J., Zosin, L.: An 8-approximation algorithm for the subset feedback vertex set problem. *SIAM J. Comput.* **30**(4), 1231–1252 (2000)
8. Fomin, F.V., Gaspers, S., Lokshtanov, D., Saurabh, S.: Exact algorithms via monotone local search. *J. ACM* **66**(2), 8:1-8:23 (2019)
9. Fomin, F.V., Heggenes, P., Kratsch, D., Papadopoulos, C., Villanger, Y.: Enumerating minimal subset feedback vertex sets. *Algorithmica* **69**(1), 216–231 (2014)
10. Fulkerson, D., Gross, O.: Incidence matrices and interval graphs. *Pacific J. Math.* **15**(3), 835–855 (1965)
11. Gavril, F.: The intersection graphs of subtrees in trees are exactly the chordal graphs. *J. Comb. Theory, Ser. B* **16**(1), 47–56 (1974)
12. Golovach, P.A., Heggenes, P., Kratsch, D., Saei, R.: Subset feedback vertex sets in chordal graphs. *J. Discrete Algorithms* **26**, 7–15 (2014)
13. Gramm, J., Guo, J., Hüffner, F., Niedermeier, R.: Data reduction and exact algorithms for clique cover. *ACM J. Exp. Algorithmics* **13** (2008)
14. Hols, E.C., Kratsch, S.: A randomized polynomial kernel for subset feedback vertex set. *Theory Comput. Syst.* **62**(1), 63–92 (2018)
15. Iwata, Y.: Linear-time kernelization for feedback vertex set. In: ICALP, pp. 68:1–68:14 (2017)
16. Iwata, Y., Kobayashi, Y.: Improved analysis of highest-degree branching for feedback vertex set. *Algorithmica* **83**(8), 2503–2520 (2021)
17. Iwata, Y., Wahlström, M., Yoshida, Y.: Half-integrality, LP-branching, and FPT algorithms. *SIAM J. Comput.* **45**(4), 1377–1411 (2016)
18. Iwata, Y., Yamaguchi, Y., Yoshida, Y.: 0/1/all CSPs, half-integral  $A$ -path packing, and linear-time FPT algorithms. In: Thorup, M. (ed.) FOCS, pp. 462–473. IEEE Computer Society (2018)
19. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W. (eds.) *Proceedings of a symposium on the Complexity of Computer Computations*, pp. 85–103. The IBM Research Symposia Series, Plenum Press, New York (1972)
20. Nemhauser, G.L., Jr., Trotter, L.E.: Properties of vertex packing and independence system polyhedra. *Math. Program.* **6**(1), 48–61 (1974)
21. Philip, G., Rajan, V., Saurabh, S., Tale, P.: Subset feedback vertex set in chordal and split graphs. *Algorithmica* **81**(9), 3586–3629 (2019)
22. Rose, D.J., Tarjan, R.E., Lueker, G.S.: Algorithmic aspects of vertex elimination on graphs. *SIAM J. Comput.* **5**(2), 266–283 (1976)
23. Stephane, F., Hammer, P.: Split graphs. In: *Proceedings of the 8th south-east Combinatorics, Graph Theory, and Computing*. pp. 311–315 (1977)
24. Thomassé, S.: A  $4k^2$  kernel for feedback vertex set. *ACM Trans. Algorithms* **6**(2), 32:1-32:8 (2010)
25. Wahlström, M.: Algorithms, measures and upper bounds for satisfiability and related problems. Ph.D. thesis, Linköping University, Sweden (2007)



26. Walter, J.R.: Representations of rigid cycle graphs. Ph.D. thesis, Wayne State University (1972)
27. Xiao, M., Nagamochi, H.: Exact algorithms for maximum independent set. *Inf. Comput.* **255**, 126–146 (2017)
28. Yannakakis, M., Gavril, F.: The maximum  $k$ -colorable subgraph problem for chordal graphs. *Inf. Process. Lett.* **24**(2), 133–137 (1987)



# An Approximation Algorithm for the $B$ -prize-collecting Multicut Problem in Trees

Xiaofei Liu<sup>1(✉)</sup> and Weidong Li<sup>2</sup>

<sup>1</sup> School of Information Science and Engineering, Yunnan University,  
Kunming, China

lxfj12016@163.com

<sup>2</sup> School of Mathematics and Statistics, Yunnan University, Kunming, China

**Abstract.** In this paper, we consider the  $B$ -prize-collecting multicut problem in trees. In this problem, we are given a tree  $T = (V, E)$ , a set of  $k$  source-sink pairs  $\mathcal{P} = \{(s_1, t_1), (s_2, t_2), \dots, (s_k, t_k)\}$  and a profit bound  $B$ . Every edge  $e \in E$  has a cost  $c_e$ , and every source-sink pair  $(s_j, t_j) \in \mathcal{P}$  has a profit  $p_j$  and a penalty  $\pi_j$ . This problem is to find a multicut  $M \subseteq E$  such that the total cost, which consists of the total cost of the edges in  $M$  and the total penalty of the pairs still connected after removing  $M$ , is minimized and the total profit of the disconnected pairs by removing  $M$  is at least  $B$ . Based on the primal-dual scheme, we present an  $(\frac{8}{3} + \epsilon)$ -approximation algorithm by carefully increasing the penalty, where  $\epsilon$  is any fixed positive number.

**Keywords:** Multicut problem in trees ·  $B$ -prize-collecting ·  
Approximation algorithm · Primal-dual scheme

## 1 Introduction

The minimum multicut problem in trees is a special case of the minimum multicut problem proposed by [9], which has been found in many applications [17]. Given a tree  $T = (V, E)$  and a set of  $k$  source-sink pairs  $\mathcal{P} = \{(s_1, t_1), (s_2, t_2), \dots, (s_k, t_k)\}$ , each edge  $e$  has a nonnegative cost  $c_e$ . The minimum multicut problem in trees is to find a multicut  $M \subseteq E$  such that all pairs in  $\mathcal{P}$  are disconnected by removing  $M$  from  $T$  and the total cost of the edges is minimized. Garg et al. [4] presented a 2-approximation algorithm based on the primal-dual technique. Then Levin and Segev [11] presented a 2-approximation algorithm based on the linear programming rounding. Recently, Guo and Niedermeier [6] and Galby et al. [3] proved that this problem is fixed-parameter tractable, respectively.

In many cases, disconnecting all pairs may not be a good strategy. To guarantee the number of disconnected pairs, Levin and Segev [11] considered the partial multicut problem in trees, which is a generalization of the minimum multicut problem in trees. In this problem, instead of the requirement that all pairs must

be disconnected, only  $l(\leq k)$  pairs must be disconnected. Levin and Segev [11] presented an  $(\frac{8}{3} + \epsilon)$ -approximation algorithm based on the Lagrangian relaxation technique for any fixed  $\epsilon > 0$ . Then, Könemann et al. [10] presented a unified approach for partial covering problems, which is a generalization of the partial multicut problem in trees, and the approximation factor of this approach for the partial multicut problem in trees is  $(\frac{8}{3} + \epsilon)$ .

Sometimes we need to pay the penalty for the pairs still connected after removing the multicut. Levin and Segev [11] considered the prize-collecting multicut problem in trees, which is another generalization of the minimum multicut problem in trees. In this problem, the pair can be connected by removing the multicut, however, we need to pay a penalty. This problem is to find a multicut  $M \subseteq E$  such that the total cost of the edges in  $M$  plus the total penalty of the pairs still connected after removing  $M$  is minimized. Segev [11] presented a 2-approximation algorithm based on a primal-dual scheme. Liu and Li [14] presented a 3-approximation algorithm for the prize-collecting multicut problem with submodular penalties, where the penalty is determined by a submodular function.

Combining the partial problem and the prize-collecting problem, the  $k$ -prize-collecting problems are considered and have gradually become a research hotspot in the field of theoretical computers. Hou et al. [8] considered the  $k$ -prize-collecting multicut problem in trees. This problem is to find a multicut  $M \subseteq E$  such that at least  $k$  pairs in  $\mathcal{P}$  are disconnected by removing  $M$  and the total cost of edges in  $M$  plus the total penalty of the pairs still connected after removing  $M$  is minimized. They presented a  $(4 + \epsilon)$ -approximation algorithm based on the primal-dual and Lagrangian relaxation techniques, where  $\epsilon$  is any fixed positive number. Liu et al. [16] considered the  $k$ -prize-collecting minimum vertex cover problem, which is a special case of the  $k$ -prize-collecting multicut problem in trees, and presented a 2-approximation algorithm. Han et al. [7] considered the  $k$ -prize-collecting Steiner tree problem, and presented a 5-approximation algorithm. Liu et al. [15] considered the  $k$ -prize-collecting power cover problem, and presented a  $3^\alpha$ -approximation algorithm, where  $\alpha$  is the attenuation factor. Then, Liu et al. [13] presented a  $5 \cdot 2^\alpha$ -approximation algorithm for this problem on the plane.

In the real world, pairs may have different priorities which can be described as profits, and Gao et al. [5] consider the  $B$ -prize-collecting set cover problem, where each element has a profit and the objective is to find a minimum subcollection such that the total profit of the covered element is at least  $B$  and the total cost of the subcollection plus the total penalty of the uncovered elements is minimized. They presented a  $(2f + \epsilon)$ -approximation algorithm, where  $f$  is the maximum frequency of an element,  $\epsilon$  is any fixed positive number. In this paper, we consider the  $B$ -prize-collecting multicut problem in trees, which is a special case of the  $B$ -prize-collecting set cover problem, and present an  $(\frac{8}{3} + \epsilon)$ -approximation algorithm by carefully increasing the penalty based on the primal-dual scheme, where  $\epsilon$  is any fixed positive number.

The rest of the paper is organized as follows. In the second part, we formally introduce the  $B$ -prize-collecting multicut problem in trees. In the third section, we present the approximation algorithm for the  $B$ -prize-collecting multicut problem in trees. Finally, we give a brief conclusion.

## 2 Preliminaries

We are given a tree  $T = (V, E)$ , a set of  $k$  source-sink pairs  $\mathcal{P} = \{(s_1, t_1), (s_2, t_2), \dots, (s_k, t_k)\}$  and a profit bound  $B$ , where  $V = \{v_1, v_2, \dots, v_n\}$ ,  $E = \{e_1, e_2, \dots, e_{n-1}\}$ ,  $s_j, t_j \in V$  for any  $j \in \{1, 2, \dots, k\}$ . We assume that

$$\sum_{j:(s_j, t_j) \in \mathcal{P}} p_j \geq B.$$

Every edge  $e$  in  $E$  has a nonnegative cost  $c_e$ , and every source-sink pair  $(s_j, t_j)$  in  $\mathcal{P}$  has a nonnegative profit  $p_j$  and a nonnegative penalty  $\pi_j$ . The  $B$ -prize-collecting multicut problem in trees ( $B$ -PCMT) is to find a multicut  $M \subseteq E$  and a set  $R \subseteq \mathcal{P}$  of the pairs still connected after removing  $M$  such that the total cost, which consists of the cost of the edges in  $M$  and the penalty cost of the pairs in  $R$ , is minimized and the disconnected profit of the pairs by removing  $M$  is at least  $B$ , i.e.,  $\sum_{j:(s_j, t_j) \in \mathcal{P} \setminus R} p_j \geq B$ .

When  $p_j = 1$  for any  $(s_j, t_j) \in \mathcal{P}$ , the  $B$ -prize-collecting multicut problem is exactly the  $k$ -prize-collecting multicut problem in trees [8]. When  $p_j = 1$  and  $\pi_j = 0$  for any  $(s_j, t_j) \in \mathcal{P}$ , the  $B$ -prize-collecting multicut problem is exactly the partial multicut problem in trees [11]. When  $B = 0$ , the  $B$ -prize-collecting multicut problem is exactly the prize-collecting multicut problem in trees [11]. When  $B = \sum_{j:(s_j, t_j) \in \mathcal{P}} p_j$ , the  $B$ -prize-collecting multicut problem is exactly the multicut problem in trees [4].

Let  $(M^*, R^*)$  be an optimal solution for the  $B$ -PCMT, and its objective value is  $OPT$ , where  $R^*$  is the set of the pairs still connected after removing  $M^*$ . Inspired by the preprocessing step in [11], given a parameter  $\epsilon > 0$ , we can assume that any edge  $e \in E$  satisfies

$$c_e \leq \epsilon \cdot OPT. \tag{1}$$

## 3 The Increasing Penalty Algorithm

The  $B$ -PCMT is a generalization of the prize-collecting multicut problem in trees (PCMT), and there is a 2-approximation algorithm [11], denoted as Algorithm  $\mathbb{A}$ , to solve the PCMT. We cannot directly implement Algorithm  $\mathbb{A}$  to solve the  $B$ -PCMT, since its output solution cannot ensure that the disconnected profit is at least  $B$ . The reason is that it is better to pay the penalty with some pairs than to make it disconnected by removing some edges. Based on this observation, by carefully increasing the penalty of the pairs, we construct an auxiliary instance of the PCMT. Using Algorithm  $\mathbb{A}$  to solve this auxiliary instance, we can obtain a feasible solution of the instance of the  $B$ -PCMT.

We are given an instance  $T$  for  $B$ -PCMT, in which  $V$  is the vertex set,  $E$  is the edge set and  $\mathcal{P}$  is a source-sink pair set, where every edge  $e$  has a cost  $c_e$  and every pair  $(s_j, t_j)$  has a profit  $p_j$  and a penalty  $\pi_j$ . Given a parameter  $\lambda \geq 0$ , we construct an auxiliary penalty  $\pi_{\lambda}$ , where

$$\pi_{\lambda, j} = \lambda \cdot p_j + \pi_j, \forall (s_j, t_j) \in \mathcal{P}.$$

Then, for any  $\lambda \geq 0$  and instance  $T$ , the auxiliary instance  $T_\lambda$  of the PCMT consists of the vertex set  $V$ , the edge set  $E$  and the source-sink pair set  $\mathcal{P}$ , where every edge  $e$  has a cost  $c_e$  and every pair  $(s_j, t_j)$  has a penalty  $\pi_{\lambda, j}$ .

Based on the proof of Theorem 3 in [11], we have the following lemma.

**Lemma 1.** [11] *For any instance  $T_\lambda$  of the PCMT, let  $(M_\lambda, R_\lambda)$  be the output solution generated by Algorithm  $\mathbb{A}$ , and we have*

$$\sum_{e \in M_\lambda} c_e + 2 \sum_{j: (s_j, t_j) \in R_\lambda} \pi_{\lambda, j} \leq 2 \cdot OPT_\lambda,$$

where  $OPT_\lambda$  is the optimal value of instance  $T_\lambda$ .

Then, we introduce a method for finding a feasible solution for instance  $T$  by Algorithm  $\mathbb{A}$  and carefully increasing the penalty, and the algorithm consists of three steps. For convenience, for any  $\lambda \geq 0$  and  $(M_\lambda, R_\lambda)$  generated by Algorithm  $\mathbb{A}$ , let

$$P_\lambda = \sum_{j: (s_j, t_j) \in \mathcal{P} \setminus R_\lambda} p_j$$

be the disconnected profit of the pairs by removing  $M_\lambda$ .

(1). Using a binary search over the interval  $[0, \frac{1}{\min_j p_j} \sum_{e \in E} c_e + 1]$  and Algorithm  $\mathbb{A}$ , we can find  $\lambda_1$  and  $\lambda_2$  satisfying  $\lambda_1 \geq \lambda_2$ ,  $\lambda_1 - \lambda_2 \leq \frac{\epsilon c_{\min}}{P_P}$ , and  $P_{\lambda_1} \geq B \geq P_{\lambda_2}$ , where  $c_{\min} = \min_{e \in E} c_e$ ,

$$P_P = \sum_{j: (s_j, t_j) \in \mathcal{P}} p_j.$$

In particular, if  $P_0 \geq P$ ,  $P_{\lambda_1} = B$  or  $P_{\lambda_2} = B$ , then  $P_0$ ,  $P_{\lambda_1}$  or  $P_{\lambda_2}$  is output and the algorithm stops.

(2). We construct an additional feasible solution  $(M_a, R_a)$ , where  $M_a$  is constructed by augmenting  $M_{\lambda_2}$  with a carefully chosen subset  $M'_{\lambda_1} \subseteq M_{\lambda_1}$ , and  $R_a$  is the set of the pairs still connected after removing  $M_a$ .

(3). The minimum solution between  $(M_{\lambda_1}, R_{\lambda_1})$  and  $(M_a, R_a)$  is output, and the algorithm stops. We define  $w(M, R)$  to denote the objective value of  $(M, R)$ ; *i.e.*,

$$w(M, R) = \sum_{e \in M} c_e + \sum_{j: (s_j, t_j) \in R} \pi_j.$$

Then, we illustrate the method for constructing  $M'_{\lambda_1}$ . First, each pair  $(s_j, t_j) \in (U \setminus R_{\lambda_1}) \cap R_{\lambda_2}$  is assigned to an arbitrary set in  $M_{\lambda_1} \setminus M_{\lambda_2}$  that contains it, and  $\varphi(e)$  denotes the total profit of the disconnected pairs assigned by removing  $e$ . Then, the edges are sorted with  $\varphi(e) > 0$  in  $M_{\lambda_1} \setminus M_{\lambda_2}$  in nondecreasing order of  $\frac{c_e}{\varphi(e)}$ . Finally, let  $M'_{\lambda_1} = \{e_1, \dots, e_q\}$ , where  $q$  is the minimal index for which  $\sum_{i=1}^q \varphi(e_i) \geq B - P_{\lambda_2}$ .

**Lemma 2.**  $\sum_{e \in M'_{\lambda_1}} c_e \leq \frac{B - P_{\lambda_2}}{P_{\lambda_1} - P_{\lambda_2}} \sum_{e \in M_{\lambda_1} \setminus M_{\lambda_2}} c_e + \epsilon \cdot OPT.$

*Proof.* Let  $k = |\{e \in M_{\lambda_1} \setminus M_{\lambda_2} | \varphi(e) > 0\}|$ ; then,  $\frac{c_{e_1}}{\varphi(e_1)} \leq \frac{c_{e_2}}{\varphi(e_2)} \leq \dots \leq \frac{c_{e_k}}{\varphi(e_k)}$  and  $\frac{\sum_{i=1}^{q-1} c_{e_i}}{\sum_{i=1}^{q-1} \varphi(e_i)} \leq \frac{\sum_{i'=1}^k c_{e_{i'}}$ ; *i.e.*,

$$\sum_{i=1}^{q-1} c_{e_i} \leq \frac{\sum_{i=1}^{q-1} \varphi(e_i)}{\sum_{i'=1}^k \varphi(e_{i'})} \sum_{i'=1}^k c_{e_{i'}} < \frac{B - P_{\lambda_2}}{P_{\lambda_1} - P_{\lambda_2}} \cdot \sum_{e: e \in M_{\lambda_1} \setminus M_{\lambda_2}} c_e,$$

where the second inequality follows from  $\sum_{i=1}^{q-1} \varphi(e_i) < B - P_{\lambda_2}$  and  $\{e_1, \dots, e_k\} \subseteq M_{\lambda_1} \setminus M_{\lambda_2}$ .

By inequality (1), the cost of each edge in  $E$  is at most  $\epsilon \cdot OPT$ ; *i.e.*,

$$\sum_{i=1}^q c_{e_i} = \sum_{i=1}^{q-1} c_{e_i} + c_{e_q} \leq \frac{B - P_{\lambda_2}}{P_{\lambda_1} - P_{\lambda_2}} \sum_{e: e \in M_{\lambda_1} \setminus M_{\lambda_2}} c_e + \epsilon \cdot OPT.$$

□

We propose the detailed primal-dual algorithm, which is shown in Algorithm 1.

**Lemma 3.** For any  $\lambda \geq 0$ , let  $(M_\lambda, R_\lambda)$  be the output solution generated by Algorithm A on instance  $T_\lambda$ , and we have

$$\sum_{e: e \in M_\lambda} c_e + \sum_{j: (s_j, t_j) \in R_\lambda} \pi_j \leq 2 \cdot OPT + 2\lambda \cdot (P_\lambda - B),$$

where  $OPT$  is the optimal value of the  $B$ -PCMT on instance  $T$  and  $P_\lambda = \sum_{j: (s_j, t_j) \in \mathcal{P} \setminus R_\lambda} p_j$  is the disconnected profit of the pairs by removing  $M_\lambda$ .

*Proof.* Let  $(M^*, R^*)$  be an optimal solution of the  $B$ -PCMT on instance  $T$ ; then, for any  $\lambda \geq 0$ ,  $(M^*, R^*)$  is also a feasible solution of  $T_\lambda$ , and

$$\begin{aligned} OPT_\lambda &\leq \sum_{e: e \in M^*} c_e + \sum_{j: (s_j, t_j) \in R^*} (\lambda \cdot p_j + \pi_j) \\ &= OPT + \lambda \cdot \sum_{j: (s_j, t_j) \in R^*} p_j \\ &\leq OPT + \lambda \cdot (P_{\mathcal{P}} - B), \end{aligned} \tag{2}$$

where  $OPT_\lambda$  is the optimal value of the PCMT on instance  $T_\lambda$  and  $P_{\mathcal{P}} = \sum_{j: (s_j, t_j) \in \mathcal{P}} p_j$ , and the second inequality follows from  $\sum_{j: (s_j, t_j) \in \mathcal{P} \setminus R^*} p_j \geq B$ .

---

**Algorithm 1:** Increasing penalty algorithm

---

**Input:** An instance  $T$  of  $B$ -PCMT.

**Output:** A feasible solution  $(M, R)$ .

- 1 Set  $\lambda_1 := \frac{1}{\min_j p_j} \sum_{e:e \in E} c_e$  and  $\lambda_2 := 0$ .
  - 2 Construct the instance  $T_{\lambda_1}$  and  $T_{\lambda_2}$  of the PCMT, and let  $(M_{\lambda_1}, R_{\lambda_1})$  and  $(M_{\lambda_2}, R_{\lambda_2})$  be the output solutions generated by Algorithm **A** on instance  $T_{\lambda_1}$  and  $T_{\lambda_2}$ , respectively.
  - 3 If  $\sum_{j:(s_j, t_j) \in \mathcal{P} \setminus R_{\lambda_1}} p_j = B$ , let  $(M, R) := (M_{\lambda_1}, R_{\lambda_1})$  and go to **16**; If  $\sum_{j:(s_j, t_j) \in \mathcal{P} \setminus R_{\lambda_2}} p_j \geq B$ , let  $(M, R) := (M_{\lambda_2}, R_{\lambda_2})$  and go to **16**.
  - 4 **while**  $\lambda_1 - \lambda_2 \leq \frac{c_{\min}}{P_{\mathcal{P}}}$  **do**
  - 5     Set  $\lambda' = \frac{\lambda_1 + \lambda_2}{2}$  and construct the instance  $T_{\lambda'}$  of the PCMT. Let  $(M_{\lambda'}, R_{\lambda'})$  be the output solution generated by Algorithm **A** on instance  $T_{\lambda'}$ .
  - 6     **if**  $\sum_{j:(s_j, t_j) \in \mathcal{P} \setminus R_{\lambda'}} p_j = B$  **then**
  - 7         Let  $(M, R) := (M_{\lambda'}, R_{\lambda'})$  and go to **16**;
  - 8     **else if**  $\sum_{j:(s_j, t_j) \in \mathcal{P} \setminus R_{\lambda'}} p_j > B$  **then**
  - 9         Set  $\lambda_1 := \lambda'$
  - 10    **else if**  $\sum_{j:(s_j, t_j) \in \mathcal{P} \setminus R_{\lambda'}} p_j < B$  **then**
  - 11         Set  $\lambda_2 := \lambda'$ .
  - 12 Construct function  $\varphi : M_{\lambda_1} \setminus M_{\lambda_2} \rightarrow \mathbb{R}_{\geq 0}$  as above, and sort the edges in  $M_{\lambda_1} \setminus M_{\lambda_2}$  such that  $\frac{c_{e_1}}{\varphi(e_1)} \leq \frac{c_{e_2}}{\varphi(e_2)} \leq \dots$ . Set  $M'_{\lambda_1} := \emptyset$  and  $i := 1$ .
  - 13 **while**  $\sum_{e:e \in M'_{\lambda_1}} \varphi(e) \geq B - \sum_{e:e \in M'_{\lambda_2}} \varphi(e)$  **do**
  - 14     set  $M'_{\lambda_1} := M'_{\lambda_1} \cup \{e_i\}$  and  $i := i + 1$ .
  - 15 Let  $M_a := M'_{\lambda_1} \cup M_{\lambda_2}$  and  $R_a$  be a set of the pairs still connected after removing  $M_a$ . Let  $(M, R) := \arg \min\{w(M_{\lambda_1}, R_{\lambda_1}), w(M_a, R_a)\}$ .
  - 16 Output  $(M, R)$ .
- 

Let  $(M_\lambda, R_\lambda)$  be the output solution generated by Algorithm **A** on instance  $T_\lambda$  and

$$\begin{aligned}
 & \sum_{e:e \in M_\lambda} c_e + \sum_{j:(s_j, t_j) \in R_\lambda} \pi_j \\
 = & \sum_{e:e \in M_\lambda} c_e + \sum_{j:(s_j, t_j) \in R_\lambda} (\lambda \cdot p_j + \pi_j) - \sum_{j:(s_j, t_j) \in R_\lambda} \lambda \cdot p_j \\
 \leq & 2 \cdot OPT_\lambda - \sum_{j:(s_j, t_j) \in R_\lambda} (\lambda \cdot p_j + \pi_j) - \sum_{j:(s_j, t_j) \in R_\lambda} \lambda \cdot p_j \\
 \leq & 2 \cdot OPT_\lambda - 2\lambda \cdot (P_{\mathcal{P}} - P_\lambda) \\
 \leq & 2OPT + 2\lambda \cdot (P_\lambda - B),
 \end{aligned}$$

where the first inequality follows from Lemma 1, the second inequality follows from  $\pi_j \geq 0$  for any  $(s_j, t_j) \in \mathcal{P}$ , and the last inequality follows from inequality (2). □

**Theorem 1.** *Let  $w(M, R)$  be the objective value of  $(M, R)$  for the B-PCMT on instance  $T$ ; then,  $w(M, R) \leq (\frac{8}{3} + 5\epsilon) \cdot OPT$ , where  $(M, R)$  is generated by Algorithm 1.*

*Proof.* If  $\sum_{j:(s_j, t_j) \in U \setminus R_0} p_j \geq B$ , then  $(M_0, R_0)$  is a feasible solution for the B-PCMT on instance  $T$  and  $(M, R) = (M_0, R_0)$ . Thus,

$$w(M, R) = w(M_0, R_0) \leq 2OPT + 2 \cdot 0(P_0 - B) = 2 \cdot OPT,$$

where the inequality follows from Lemma 3.

If  $P_{\lambda_1} = B$ ,  $(M_{\lambda_1}, R_{\lambda_1})$  is a feasible solution for the B-PCMT on instance  $T$ , and  $(M, R) = (M_{\lambda_1}, R_{\lambda_1})$ . This means that

$$w(M, R) = w(M_{\lambda_1}, R_{\lambda_1}) \leq 2OPT_{\lambda_1} - 2\lambda_1 \cdot (P_{\lambda_1} - B) = 2 \cdot OPT,$$

where the first inequality follows from Lemma 3.

If  $P_{\lambda_2} = B$ ; similarly,  $w(M, R) \leq 2 \cdot OPT$ .

Then, we consider the case with

$$P_{\lambda_1} > B, \text{ and } P_{\lambda_2} < B.$$

Let  $\alpha = \frac{B - P_{\lambda_2}}{P_{\lambda_1} - P_{\lambda_2}}$ , and we have

$$\begin{aligned} & \alpha \cdot w(M_{\lambda_1}, R_{\lambda_1}) + (1 - \alpha) \cdot w(M_{\lambda_2}, R_{\lambda_2}) \\ & \leq 2\alpha(OPT + \lambda_1(P_{\lambda_1} - B)) + 2(1 - \alpha)(OPT + \lambda_2(P_{\lambda_2} - B)) \\ & \leq 2OPT + 2\alpha(\lambda_2 + \frac{\epsilon c_{\min}}{P_{\mathcal{P}}})(P_{\lambda_1} - B) + 2(1 - \alpha)\lambda_2(P_{\lambda_2} - B) \\ & = 2OPT + 2\lambda_2(\alpha(P_{\lambda_1} - B) + (1 - \alpha)(P_{\lambda_2} - B)) + 2\alpha\epsilon c_{\min} \frac{P_{\lambda_1} - B}{P_{\mathcal{P}}} \\ & \leq 2OPT + 2\epsilon c_{\min} \leq 2(1 + \epsilon) \cdot OPT, \end{aligned} \tag{3}$$

where  $P_{\mathcal{P}} = \sum_{j:(s_j, t_j) \in \mathcal{P}} p_j$ ; the first inequality follows from Lemma 3; the second inequality follows from  $\lambda_1 - \lambda_2 \leq \frac{\epsilon c_{\min}}{P_{\mathcal{P}}}$  and  $\alpha \in (0, 1)$  by  $P_{\lambda_1} > B$ ; and the third inequality follows from  $\alpha(P_{\lambda_1} - B) + (1 - \alpha)(P_{\lambda_2} - B) = 0$  by  $\alpha = \frac{B - P_{\lambda_2}}{P_{\lambda_1} - P_{\lambda_2}}$ ,  $\alpha \in (0, 1)$  and  $P_{\mathcal{P}} \geq P_{\lambda_1} - B$ .

This implies that the objective value of  $(M_{\lambda_1}, R_{\lambda_1})$  is

$$w(M_{\lambda_1}, R_{\lambda_1}) \leq \frac{2(1 + \epsilon) \cdot OPT - (1 - \alpha) \cdot w(M_{\lambda_2}, R_{\lambda_2})}{\alpha}.$$

**Case. 1.**  $\alpha \cdot \frac{w(M_{\lambda_2}, R_{\lambda_2})}{OPT} \geq \frac{2}{3}$ . It is not hard to obtain that

$$0 \leq \frac{w(M_{\lambda_2}, R_{\lambda_2})}{OPT} \leq 2.$$



This means that

$$\begin{aligned}
 w(M_{\lambda_1}, R_{\lambda_1}) &\leq \frac{2(1 + \epsilon) \cdot OPT - (1 - \alpha) \cdot w(M_{\lambda_2}, R_{\lambda_2})}{\alpha} \\
 &\leq \frac{2(1 + \epsilon) \cdot OPT - (1 - \alpha) \cdot \frac{2}{3\alpha} \cdot OPT}{\alpha} \\
 &= \left(-\frac{2}{3} \cdot \left(\frac{1}{\alpha}\right)^2 + \left(\frac{8}{3} + 2\epsilon\right) \cdot \left(\frac{1}{\alpha}\right)\right) \cdot OPT \\
 &\leq \left(\frac{8}{3} + 5\epsilon\right) \cdot OPT,
 \end{aligned}$$

where the second inequality follows from  $\alpha \cdot \frac{w(M_{\lambda_2}, R_{\lambda_2})}{OPT} \geq \frac{2}{3}$  and the last inequality follows from  $\frac{1}{\alpha} \in [1, +\infty)$  and the fact that function  $f(x) = -\frac{2}{3} \cdot x^2 + \left(\frac{8}{3} + 2\epsilon\right) \cdot x$  satisfies  $\frac{8}{3} + 4\epsilon \geq f\left(2 + \frac{3}{2}\epsilon\right) \geq f(x')$  for any  $x' \in [1, +\infty)$ .

**Case. 2.**  $\alpha \cdot \frac{w(M_{\lambda_2}, R_{\lambda_2})}{OPT} < \frac{2}{3}$ ; *i.e.*,

$$\alpha \cdot w(M_{\lambda_2}, R_{\lambda_2}) < \frac{2}{3} \cdot OPT.$$

Since  $M_a = M'_{\lambda_1} \cup M_{\lambda_2}$ , we have  $R_a \subseteq R_{\lambda_2}$ , and the objective value of  $(M_a, R_a)$  is

$$\begin{aligned}
 w(M_a, R_a) &= \sum_{e: e \in M_{\lambda_2}} c_e + \sum_{e: e \in M'_{\lambda_1}} c_e + \sum_{j: (s_j, t_j) \in R_a} \pi_j \\
 &\leq w(M_{\lambda_2}, R_{\lambda_2}) + \frac{B - P_{\lambda_2}}{P_{\lambda_1} - P_{\lambda_2}} \sum_{e: e \in M_{\lambda_1} \setminus M_{\lambda_2}} c_e + \epsilon \cdot OPT \\
 &\leq (1 - \alpha) \cdot w(M_{\lambda_2}, R_{\lambda_2}) + \alpha \cdot w(M_{\lambda_2}, R_{\lambda_2}) \\
 &\quad + \alpha \cdot w(M_{\lambda_1}, R_{\lambda_1}) + \epsilon \cdot OPT \\
 &\leq 2 \cdot (1 + \epsilon) \cdot OPT + \alpha \cdot w(M_{\lambda_2}, R_{\lambda_2}) + \epsilon \cdot OPT \\
 &< 2 \cdot (1 + \epsilon) \cdot OPT + \frac{2}{3} OPT + \epsilon \cdot OPT \\
 &= \left(\frac{8}{3} + 3\epsilon\right) \cdot OPT,
 \end{aligned}$$

where the first inequality follows from  $R_a \subseteq R_{\lambda_2}$  and  $\pi_j \geq 0$  for any  $(s_j, t_j) \in \mathcal{P}$ , the second inequality follows from Lemma 2, the third inequality follows from inequality (3), and the last inequality follows from  $\alpha \cdot w(M_{\lambda_2}, R_{\lambda_2}) < \frac{2}{3} \cdot OPT$ .

Therefore, the theorem holds.  $\square$

## 4 Conclusions and Future Work

We introduce the  $B$ -prize-collecting multicut problem in trees, which is a generalization of the minimum multicut problem in trees [4], the partial multicut problem in trees [11], the prize-collecting multicut problem in trees [11], and

the  $k$ -prize-collecting multicut problem in trees [8]. Based on the primal-dual scheme, we present an  $(\frac{8}{3} + \epsilon)$ -approximation algorithm by carefully increasing the penalty, where  $\epsilon$  is any fixed positive number.

Recently, problems with submodular penalties have gradually become a research hotspot in the field of theoretical computers and combinatorial optimization, and the submodular penalty version of this problem is worth considering, in which the penalty is determined by a submodular function.

**Acknowledgement.** The work is supported in part by the National Natural Science Foundation of China [No. 12071417].

**Declaration.** The authors declare that they have no known competing financial interests.

## References

1. Du, D., Lu, R., Xu, D.: A primal-dual approximation algorithm for the facility location problem with submodular penalties. *Algorithmica* **63**(1–2), 191–200 (2012)
2. Garey, M.R., Johnson, D.S.: *Computers and intractability: a guide to the theory of NP-completeness*. W.H Freeman and Company, New York (1990)
3. Galby, E., Marx, Dániel., Schepper, P., Sharma, R., Tale, P.: Parameterized complexity of weighted multicut in trees. [arXiv:2205.10105](https://arxiv.org/abs/2205.10105) (2022) <https://doi.org/10.48550/arXiv.2205.10105>
4. Garg, N., Vazirani, V.V., Yannakakis, M.: Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica* **18**, 3–20 (1997)
5. Guo, J., Liu, W., Hou, B.: An approximation algorithm for P-prize-collecting set cover problem. *J. Oper. Res. Soc. China* (2021) <https://doi.org/10.1007/s40305-021-00364-7>
6. Guo, J., Niedermeier, R.: Exact algorithms and applications for tree-like weighted set cover. *J. Discrete Algorithms* **4**(4), 608–622 (2006)
7. Han, L., Xu, D., Du, D., Wu, C.: A 5-approximation algorithm for the  $k$ -prize-collecting Steiner tree problem. *Opti. Lett.s* **13**, 573–585 (2019)
8. Hou, X., Liu, W., Hou, B.: An approximation algorithm for the  $k$ -prize-collecting multicut on a tree problem. *Theoret. Comput. Sci.* **844**, 26–33 (2020)
9. Hu, T.C.: *Integer Programming and Network Flows*. Princeton University Press, Princeton (1969)
10. Könemann, J., Parekh, O., Segev, D.: A unified approach to approximating partial covering problems. *Algorithmica* **59**(4), 489–509 (2011)
11. Levin, A., Segev, D.: Partial multicuts in trees. *Theoret. Comput. Sci.* **369**(1–3), 384–395 (2006)
12. Li, Y., Du, D., Xiu, N., Xu, D.: Improved approximation algorithms for the facility location problems with linear /submodular penalties. *Algorithmica* **73**, 460–482 (2015)
13. Liu, X., Dai, H., Li, S., Li, W.:  $k$ -prize-collecting minimum power cover problem with submodular penalties on a plane (in Chinese). *Scientia Sinica Informationis* **52**(6), 947 (2022)
14. Liu, X., Li, W.: Combinatorial approximation algorithms for the submodular multicut problem in trees with submodular penalties. *J. Comb. Opt.* **44**, 1964–1976 (2022). <https://doi.org/10.1007/s10878-020-00568-2>

15. Liu, X., Li, W., Xie, R.: A primal-dual approximation algorithm for the  $k$ -prize-collecting minimum power cover problem. *Opt. Lett.* **16**, 2373–2385 (2022). <https://doi.org/10.1007/s10878-020-00568-2>
16. Liu, X., Li, W., Yang, J.: A primal-dual approximation algorithm for the  $k$ -prize-collecting minimum vertex cover problem with submodular penalties. *Front. Comput. Sci.* **17**(3), 173404 (2023). <https://doi.org/10.1007/s11704-022-1665-9>
17. Zhang, P., Zhu, D., Luan, J.: An approximation algorithm for the generalized  $k$ -multicut problem. *Discret. Appl. Math.* **160**(7–8), 1240–1247 (2012)



# Two-Stage Non-submodular Maximization

Hong Chang<sup>1</sup>, Zhicheng Liu<sup>2</sup>, Ping Li<sup>3</sup>, and Xiaoyan Zhang<sup>1</sup>(✉)

<sup>1</sup> School of Mathematical Science and Institute of Mathematics,  
Nanjing Normal University, Nanjing 210023, China

{changh,zhangxiaoyan}@njnu.edu.cn, royxyzhang@gmail.com

<sup>2</sup> Beijing Institute for Scientific and Engineering Computing,  
Beijing University of Technology, Beijing 100124, China

<sup>3</sup> Shandong Qiguang Information Technology Co. Ltd., Jiading Qu 253000, China  
liping@vista.aero

**Abstract.** The concept of submodularity finds wide applications in data science, artificial intelligence, and machine learning, providing a boost to the investigation of new ideas, innovative techniques, and creative algorithms to solve different submodular optimization problems arising from a diversity of applications. However pure submodular problems only represent a small portion of the problems we are facing in real life applications. To solve these optimization problems, an important research method is to describe the characteristics of the non-submodular functions. The non-submodular functions is a hot research topic in the study of nonlinear combinatorial optimizations. In this paper, we combine and generalize the curvature and the generic submodularity ratio to design an approximation algorithm for two-stage non-submodular maximization under a matroid constraint.

**Keywords:** Two-stage submodular maximization · Matroid constraint · Curvature · Generic submodularity ratio

## 1 Introduction

Submodular function maximization has drawn much attention practical and theoretical interests [5, 6, 11, 13]. For a given set  $V$ , the function  $f : 2^V \rightarrow \mathbb{R}$  is said to be submodular if  $f(X) + f(Y) \geq f(X \cap Y) + f(X \cup Y)$  for  $\forall X, Y \subseteq V$ . A set function  $f$  is called monotone if  $f(X) \leq f(Y)$  for all  $X \subseteq Y \subseteq V$  and it is said to be normalized when  $f(\emptyset) = 0$ . The well-known greedy algorithm presents a constraint-factor approximation ratio  $1 - 1/e$  for submodular maximization subject to a cardinality constraint [10]. The bounds can be improved if one make further assumptions on submodular functions. For example, the curvature of a submodular function  $f$  in [3] is defined as

$$k_f = 1 - \min_{v \in V} \frac{f(V) - f(V \setminus \{v\})}{f(v)},$$

and noting that the curvature is computable with a linear number of function oracle calls, then the greedy algorithm obtains  $\frac{1}{k_f}(1 - e^{-k_f})$  guarantee under a cardinality constraint [3].

The greedy algorithm is a simple and effective technique to solve many optimization problems. However, the ground set is often so large that the well-known greedy algorithm is not enough efficient. One solution to the problem is to give some training functions to reduce the ground set, and then Balkanski et al. [1] gave the concept of the two-stage submodular maximization problem. For a ground set  $V$  and a constant  $k$ , the objective is to obtain a set  $S \subseteq V$  of size at most  $k$  and  $m$  subsets  $T_1, T_2, \dots, T_m$  in  $\mathcal{I}(S)$  to maximize the following

$$\frac{1}{m} \sum_{i=1}^m \max_{T \in \mathcal{I}(S)} f_i(T),$$

where  $f_i : 2^V \rightarrow \mathbb{R}_+$  is submodular for  $i = 1, \dots, m$ , and  $\mathcal{I}(S)$  is a constraint set over the reduced ground set  $S \subseteq V$ .

Related works have been conducted in the area of the two-stage submodular maximization. For a cardinality constraint, that is  $\mathcal{I}(S) = \{T : |T| \leq k\}$ . When  $k$  is enough large, Balkanski et al. [1] used the continuous optimization method to design an approximation algorithm with approximation ratio, which asymptotically approaches  $1 - 1/e$ . When  $k$  is small, a local search algorithm was obtained with approximation ratio close to  $1/2$  in [1]. In addition, Mitrovic et al. [9] considered the two-stage submodular maximization with cardinality constraint under streaming and distributed settings. For a matroid constraint, Stan et al. [12] obtained a new local-search based algorithm with approximation ratio  $1 - 1/e^2$ .

On the other hand, for many applications in practice, including experimental design and sparse Gaussian processes [8], the objective function is in general not submodular. The results for submodular optimization problems are not no longer maintained. To solve these optimization problems, an important research method is to introduce some parameters to describe the characteristics of the non-submodular functions, such as submodularity ratio, curvature, generic submodularity ratio, and then design algorithms for the problems and analyze the performances of the algorithms with these parameters. Given a ground set  $V$  and a nondecreasing set function  $f : 2^V \rightarrow \mathbb{R}$ , the generic submodularity ratio of  $f$  is the largest scalar  $\gamma$  such that for any  $X \subseteq Y \subseteq V$  and any  $v \in V \setminus Y$ ,  $f(v|X) \geq \gamma f(v|Y)$ , which is a quantity characterizing how close a nonnegative nondecreasing set function is to be submodular [4]. And, a function is called  $\gamma$ -submodular if its generic submodularity ratio is  $\gamma$ . A natural curvature notion can also be introduced for non-submodular functions. We recall that the curvature of a non-negative set function [2] is the smallest scalar  $\alpha$  such that for  $\forall S, T \subseteq V, i \in S \setminus T$ ,

$$f(i|S \setminus \{i\} \cup T) \geq (1 - \alpha)f(i|S \setminus \{i\}).$$

Based on the above motivation, we discuss the two-stage  $\gamma$ -submodular maximization problem under a matroid constraint. Our main contribution is to design

an approximation algorithm with constant approximation ratio with respect to the curvature and the generic submodularity ratio. The rest of our paper is summarized as below. In Sect. 2, we show some technical preliminaries, including notations and relevant known results. In Sect. 3, we give an approximation algorithm along with its analysis. And some concluding remarks are presented in Sect. 4.

## 2 Preliminaries

Firstly, we recall the following known concepts and results for submodular functions, supermodular functions and modular function.

**Definition 1.** For a given set  $V$ , the function  $f : 2^V \rightarrow \mathbb{R}$  is called submodular if  $f(X) + f(Y) \geq f(X \cap Y) + f(X \cup Y)$  for  $\forall X, Y \subseteq V$ .

An equivalent definition is that the function  $f : 2^V \rightarrow \mathbb{R}$  is said to be submodular if  $f(e|S) \geq f(e|T)$  for  $S \subseteq T \subset V$  and  $v \in V \setminus S$ , where  $f(e|S) = f(e \cup S) - f(S)$ .

**Definition 2.** For a given set  $V$ , the function  $f : 2^V \rightarrow \mathbb{R}$  is called modular if  $f(X) + f(Y) \leq f(X \cap Y) + f(X \cup Y)$  for  $\forall X, Y \subseteq V$ .

Furthermore, we define the concept of modular functions.

**Definition 3.** For a given set  $V$ , the function  $f : 2^V \rightarrow \mathbb{R}$  is called modular if  $f(X) + f(Y) = f(X \cap Y) + f(X \cup Y)$  for  $\forall X, Y \subseteq V$ .

Next, we formally restate the two-stage submodular maximization problem. For a ground set  $V$ , and  $m$  nonnegative, monotone and normalized  $\gamma$ -submodular functions  $f_1, \dots, f_m$ , which are drawn from some unknown distribution, our aim is to select a set  $S \subseteq V$  of size at most  $k$  and  $m$  subsets  $T_1, \dots, T_m$  in  $\mathcal{I}(S)$  to maximize the following

$$\frac{1}{m} \sum_{i=1}^m \max_{T \in \mathcal{I}(S)} f_i(T), \tag{2.1}$$

where  $\mathcal{I}(S)$  is a constraint set over  $S \subseteq V$ . In our paper, the set  $\mathcal{I}(S)$  corresponds to a matroid constraint.

**Definition 4.** For a given set  $S$  and  $\mathcal{I} \in 2^S$ , a matroid  $\mathcal{M} = (S, \mathcal{I})$  satisfies three properties: (1)  $\emptyset \in \mathcal{I}$ ; (2) if  $P \subseteq Q \in \mathcal{I}$ , then  $P \in \mathcal{I}$ ; (3) if  $P, Q \in \mathcal{I}$  and  $|P| \leq |Q|$ , then  $P + q \in \mathcal{I}$ , where  $q \in Q \setminus P$ .

The mapping below is a very useful tool to study the matroid constraint, which is shown in [7].

**Proposition 1.** Let  $\mathcal{M}_i = (S, \mathcal{I}_i)$  be a matroid for  $i \in \{1, \dots, p\}$ . For  $\forall X, Y \in \mathcal{I}_i$ , there is a mapping  $\pi_i : Y \setminus X \rightarrow X \setminus Y \cup \{\emptyset\}$ , which satisfies the three properties (1)  $(X \setminus \pi_i(y)) \cup y \in \mathcal{I}_i$  for  $\forall y \in Y \setminus X$ ; (2)  $|\pi_i^{-1}(x)| \leq 1$  for  $\forall x \in X \setminus Y$ ; (3) let  $X_y = \{\pi_1(y), \dots, \pi_p(y)\}$ , then  $(X \setminus X_y) \cup y \in \cap_{i=1}^p \mathcal{I}_i$  for  $\forall y \in Y \setminus X$ .

Now, we turn to give a nice result for  $\gamma$ -submodular functions in [4].

**Proposition 2.** *Let  $f$  be a  $\gamma$ -submodular function. Then for  $\forall S \subseteq T$ ,*

$$f(T) \leq f(S) + \frac{1}{\gamma} \sum_{v \in T \setminus S} f(v|S).$$

To maximize the above objective function, we discuss the following

$$\begin{aligned} \ell_i(T) &= (1 - \alpha) \sum_{v \in T} f_i(v), \\ g_i(T) &= f_i(T) - \ell_i(T). \end{aligned}$$

where the function  $\ell_i(T)$  is modular, and the function  $g_i(T)$  is a monotone and normalized  $\gamma$ -submodular function, this is because that the function  $f_i(T)$  is monotone and normalized.

Problem (2.1) turns into the following problem: obtain a set  $S \subseteq V$  of size at most  $k$  and  $m$  subsets  $T_1, T_2, \dots, T_m$  in  $\mathcal{I}(S)$  to maximize the following

$$\frac{1}{m} \sum_{i=1}^m \max_{T \in \mathcal{I}(S)} (g_i(T) + \ell_i(T)).$$

For notational convenience, we use the following notations. In terms of the function  $g_i$ , define the marginal gain of adding an element  $x$  to the set  $T_i^j$  as

$$\Delta_i^g(x, T_i^j) = g_i(\{x\} \cup T_i^j) - g_i(T_i^j).$$

Similarly, the gain of replacing  $y$  with  $x$  in terms of the set  $T_i^j$  is represented by

$$\nabla_i^g(x, y, T_i^j) = g_i(\{x\} \cup T_i^j \setminus \{y\}) - g_i(T_i^j).$$

Furthermore, for the functions  $g_i$  and  $\ell_i$ , we define

$$\Delta_i(x, T_i^j) = \left(1 - \frac{\gamma + \frac{1}{\gamma}}{k}\right)^{k-j} \Delta_i^g(x, T_i^j) + \left(1 - \frac{1}{k}\right)^{k-j} \ell_i(x),$$

$$\nabla_i(x, y, T_i^j) = \left(1 - \frac{\gamma + \frac{1}{\gamma}}{k}\right)^{k-j} \nabla_i^g(x, y, T_i^j) + \left(1 - \frac{1}{k}\right)^{k-j} (\ell_i(x) - \ell_i(y)).$$

The set of elements in  $T_i^j$  can replace  $x$ , which will not violate the matroid constraint, is defined as

$$\mathcal{I}(x, T_i^j) = \{y \in T_i^j : T_i^j \cup \{x\} \setminus \{y\} \in \mathcal{I}(S)\}.$$

Based on the notations of  $\Delta_i(x, T_i^j)$  and  $\nabla_i(x, y, T_i^j)$ , we denote the replacement gain of  $x$  in terms of  $T_i^j$  by

$$\nabla_i(x, T_i^j) = \begin{cases} \Delta_i(x, T_i^j) & \text{if } T_i^j \cup \{x\} \in \mathcal{I}(S), \\ \max\{0, \max_{y \in \mathcal{I}(x, T_i^j)} \nabla_i(x, y, T_i^j)\} & \text{otherwise.} \end{cases}$$

In addition, we define

$$\text{Rep}_i(x, T_i^j) = \begin{cases} \emptyset & \text{if } T_i^j \cup \{x\} \in \mathcal{I}(S), \\ \arg \max_{y \in \mathcal{I}(x, T_i^j)} \nabla_i(x, y, T_i^j) & \text{otherwise.} \end{cases}$$

### 3 Problem (2.1) Under a Matroid Constraint

In this section, we discuss Problem (2.1) under  $\mathcal{I}(S)$  is a matroid constraint. A replacement greedy algorithm is shown in Sect. 3.1, and then we analyze its approximation ratio in Sect. 3.2.

#### 3.1 Algorithm

Our replacement greedy algorithm starts with  $S^0 = \emptyset$ , and runs in  $k$  rounds. In each round, a new element can be added into the current solution if it does not violate the matroid constraints or can be replaced with some element in the current set while increasing the value of the objective function.

---

#### Algorithm 1.

---

- 1:  $S^0 \leftarrow \emptyset, T_i^0 \leftarrow \emptyset (\forall 1 \leq i \leq m)$
  - 2: **for**  $1 \leq j \leq k$  **do**
  - 3:    $t^* \leftarrow \arg \max_{t \in V} \sum_{i=1}^m \nabla_i(t, T_i^{j-1})$
  - 4:    $S^j \leftarrow S^{j-1} \cup \{t^*\}$
  - 5:   **for**  $1 \leq i \leq m$  **do**
  - 6:     **if**  $\nabla_i(t^*, T_i^{j-1}) > 0$  **then**
  - 7:        $T_i^j \leftarrow T_i^{j-1} \cup \{t^*\} \setminus \text{Rep}_i(t^*, T_i^{j-1})$
  - 8:     **else**
  - 9:        $T_i^j \leftarrow T_i^{j-1}$
  - 10:    **end if**
  - 11:   **end for**
  - 12: **end for**
  - 13: Return sets  $S^k$  and  $T_1^k, T_2^k, \dots, T_m^k$
- 

#### 3.2 Theoretical Analysis

We analyze the performance guarantee of Algorithm 1, which depends on the distorted objective function as follows.

$$\Phi_j(S^j) = \sum_{i=1}^m \left( \left( 1 - \frac{\gamma + \frac{1}{\gamma}}{k} \right)^{k-j} g_i(T_i^j) + \left( 1 - \frac{1}{k} \right)^{k-j} \ell_i(T_i^j) \right).$$



**Lemma 1.** *In each iteration of Algorithm 1,*

$$\begin{aligned} & \Phi_j(S^j) - \Phi_{j-1}(S^{j-1}) \\ &= \sum_{i=1}^m \left( \nabla_i(t^j, T_i^{j-1}) + \frac{\gamma + \frac{1}{\gamma}}{k} \left( 1 - \frac{\gamma + \frac{1}{\gamma}}{k} \right)^{k-j} g_i(T_i^{j-1}) + \frac{1}{k} \left( 1 - \frac{1}{k} \right)^{k-j} \ell_i(T_i^{j-1}) \right). \end{aligned}$$

For the second term on the right side in Lemma 1, we give the lower bound in the following.

**Lemma 2.** *If the element  $t^j \in V$  is added into the current set  $S^{j-1}$ , then*

$$\sum_{i=1}^m \nabla_i(t^j, T_i^{j-1}) \geq \frac{1}{k} \sum_{i=1}^m \sum_{t \in T_i^* \setminus T_i^{j-1}} \nabla_i(t, T_i^{j-1}),$$

where  $S^* = \arg \max_{S \subseteq V, |S| \leq k} \sum_{i=1}^m \max_{T \in \mathcal{I}(S)} f_i(T)$ ,  $T_i^* = \arg \max_{A \in \mathcal{I}(S^*)} f_i(A)$ .

The following lemma is crucial to analyze the approximation ratio of Algorithm 1.

**Lemma 3.** *For  $j = 1, 2, \dots, k$ , we have*

$$\begin{aligned} & \sum_{i=1}^m \nabla_i(t^j, T_i^{j-1}) \\ & \geq \frac{1}{k} \left( 1 - \frac{\gamma + \frac{1}{\gamma}}{k} \right)^{k-j} \sum_{i=1}^m \left( \gamma(1 - \alpha)g_i(T_i^*) - \left( \gamma + \frac{1}{\gamma} \right)g_i(T_i^{j-1}) \right) \\ & \quad + \frac{1}{k} \left( 1 - \frac{1}{k} \right)^{k-j} \sum_{i=1}^m \left( \ell_i(T_i^*) - \ell_i(T_i^{j-1}) \right). \end{aligned}$$

Combining Lemma 1 and Lemma 3, the following theorem is proved as below.

**Theorem 1.** *Algorithm 1 returns a set  $S^k$  of size  $k$  such that*

$$\sum_{i=1}^m (g_i(T_i^k) + \ell_i(T_i^k)) \geq \frac{\gamma}{\gamma + \frac{1}{\gamma}} \left( 1 - e^{-(\gamma + \frac{1}{\gamma})} \right) \sum_{i=1}^m g_i(T_i^*) + (1 - e^{-1}) \sum_{i=1}^m \ell_i(T_i^*).$$

*Proof.* By the definition of the function  $\Phi$ , it is obtained that

$$\begin{aligned} & \Phi_0(S^0) = 0, \\ & \Phi_k(S^k) = \sum_{i=1}^m \left( \left( 1 - \frac{\gamma + \frac{1}{\gamma}}{k} \right)^{k-k} g_i(T_i^k) + \left( 1 - \frac{1}{k} \right)^{k-k} \ell_i(T_i^k) \right) \\ & = \sum_{i=1}^m (g_i(T_i^k) + \ell_i(T_i^k)). \end{aligned}$$

Using Lemmas 1 and Lemma 3, we have

$$\begin{aligned} & \Phi_j(S^j) - \Phi_{j-1}(S^{j-1}) \\ &= \sum_{i=1}^m \left( \nabla_i(t^j, T_i^{j-1}) + \frac{\gamma + \frac{1}{\gamma}}{k} \left( 1 - \frac{\gamma + \frac{1}{\gamma}}{k} \right)^{k-j} g_i(T_i^{j-1}) + \frac{1}{k} \left( 1 - \frac{1}{k} \right)^{k-j} \ell_i(T_i^{j-1}) \right). \end{aligned}$$

Finally,

$$\begin{aligned} & \sum_{i=1}^m (g_i(T_i^k) + \ell_i(T_i^k)) \\ &= \sum_{j=1}^k (\Phi_j(S^j) - \Phi_{j-1}(S^{j-1})) \\ &\geq \sum_{j=1}^k \left( \frac{\gamma}{k} \left( 1 - \frac{\gamma + \frac{1}{\gamma}}{k} \right)^{k-j} \sum_{i=1}^m g_i(T_i^*) + \frac{1}{k} \left( 1 - \frac{1}{k} \right)^{k-j} \sum_{i=1}^m \ell_i(T_i^*) \right) \\ &\geq \frac{\gamma}{\gamma + \frac{1}{\gamma}} \left( 1 - e^{-(\gamma + \frac{1}{\gamma})} \right) \sum_{i=1}^m g_i(T_i^*) + (1 - e^{-1}) \sum_{i=1}^m \ell_i(T_i^*). \end{aligned}$$

The curvature is an very useful assumption to obtain the following result.

**Theorem 2.** *There exists an algorithm returning a set  $S^k$  of size  $k$  such that*

$$F(S^k) \geq \left( \frac{\gamma}{\gamma + \frac{1}{\gamma}} (1 - (1 - \alpha)\gamma) (1 - e^{-(\gamma + \frac{1}{\gamma})}) + (1 - \alpha)\gamma(1 - e^{-1}) \right) OPT.$$

where  $F(S^k) = \sum_{i=1}^m \max_{T \in \mathcal{I}(S^k)} (f_i(T_i^k))$  and  $OPT$  is the optimal solution.

*Proof.* It follows from the definition of  $\ell_i(T)$  and Proposition 2 that

$$\ell_i(T) = (1 - \alpha) \sum_{v \in T} f_i(v) \geq (1 - \alpha)\gamma f_i(T).$$

Furthermore,

$$\begin{aligned} & \sum_{i=1}^m f_i(T_i^k) = \sum_{i=1}^m (g_i(T_i^k) + \ell_i(T_i^k)) \\ &\geq \frac{\gamma}{\gamma + \frac{1}{\gamma}} \left( 1 - e^{-(\gamma + \frac{1}{\gamma})} \right) \sum_{i=1}^m g_i(T_i^*) + (1 - e^{-1}) \sum_{i=1}^m \ell_i(T_i^*) \\ &= \left( \frac{\gamma}{\gamma + \frac{1}{\gamma}} (1 - e^{-(\gamma + \frac{1}{\gamma})}) \right) \sum_{i=1}^m (f_i(T_i^*) - \ell_i(T_i^*)) + (1 - e^{-1}) \sum_{i=1}^m \ell_i(T_i^*) \\ &= \left( \frac{\gamma}{\gamma + \frac{1}{\gamma}} (1 - e^{-(\gamma + \frac{1}{\gamma})}) \right) \sum_{i=1}^m f_i(T_i^*) + \left( (1 - e^{-1}) - \left( \frac{\gamma}{\gamma + \frac{1}{\gamma}} (1 - e^{-(\gamma + \frac{1}{\gamma})}) \right) \right) \sum_{i=1}^m \ell_i(T_i^*) \\ &\geq \left( \frac{\gamma}{\gamma + \frac{1}{\gamma}} (1 - e^{-(\gamma + \frac{1}{\gamma})}) \right) \sum_{i=1}^m f_i(T_i^*) + (1 - \alpha)\gamma \left( (1 - e^{-1}) - \left( \frac{\gamma}{\gamma + \frac{1}{\gamma}} (1 - e^{-(\gamma + \frac{1}{\gamma})}) \right) \right) \sum_{i=1}^m \ell_i(T_i^*) \\ &\geq \left( \frac{\gamma}{\gamma + \frac{1}{\gamma}} (1 - (1 - \alpha)\gamma) (1 - e^{-(\gamma + \frac{1}{\gamma})}) + (1 - \alpha)\gamma(1 - e^{-1}) \right) \sum_{i=1}^m f_i(T_i^*). \end{aligned}$$

Finally, we obtain that

$$F(S^k) \geq \left( \frac{\gamma}{\gamma + \frac{1}{\gamma}} (1 - (1 - \alpha)\gamma) (1 - e^{-(\gamma + \frac{1}{\gamma})}) + (1 - \alpha)\gamma(1 - e^{-1}) \right) OPT.$$

## 4 Conclusion

The objective functions for many applications in practice are in general not submodular. To solve these optimization problems, an important research method is to introduce some parameters to describe the characteristics of the non-submodular functions, such as submodularity ratio, curvature, supermodular degree, etc., and then design algorithms for the problems and analyze the performances of the algorithms with these parameters. On the other hand, it is well known that submodular maximization problem can be solved by greedy algorithms, To avoid this limitation of the regular greedy algorithm, we propose combining the distorted objective function and the greedy algorithms, which has the potential to be applicable to other optimization problems.

**Acknowledgements.** The research is supported by NSFC (Nos.12101314,12131003, 11871280,12271259,11971349), Qinglan Project, Natural Science Foundation of Jiangsu Province (No. BK20200723), and Jiangsu Province Higher Education Foundation (No.20KJB110022).

## References

1. Balkanski, E., Krause, A., Mirzsoleiman, B., Singer, Y.: Learning sparse combinatorial representations via two-stage submodular maximization. In: ICML, pp. 2207–2216 (2016)
2. Bian, A.A., Buhmann, J.M., Krause, A., Tschischek, S.: guarantees for greedy maximization of non-submodular functions with applications. In: Proceedings of ICML, pp. 498–507 (2017)
3. Conforti, M., Cornuejols, G.: Submodular set functions, matroids and the greedy algorithm: tight worst-case bounds and some generalizations of the Rado-Edmonds theorem. *Discrete Appl. Math.* **7**(3), 251–274 (1984)
4. Gong, S., Nong, Q., Sun, T., Fang, Q., Du, X., Shao, X.: Maximize a monotone function with a generic submodularity ratio. *Theor. Comput. Sci.* **853**, 16–24 (2021)
5. Krause, A., Guestrin, A.: Near-optimal nonmyopic value of information in graphical models. In: UAI, p. 5 (2005)
6. Laitila, J., Moilanen, A.: New performance guarantees for the greedy maximization of submodular set functions. *Optim. Lett.* **11**, 655–665 (2017)
7. Lee, J., Mirrokni, V.S., Nagarajan, V., Sviridenko, M.: Maximizing nonmonotone submodular functions under matroid or knapsack constraints. *SIAM J. Discrete Math.* **23**(4), 2053–2078 (2010)
8. Lawrence, N., Seeger, M., Herbrich, R.: Fast sparse gaussian process methods: the informative vector machine. *Adv. Neural. Inf. Process. Syst.* **1**(5), 625–632 (2003)

9. Mitrovic, M., Kazemi, E., Zadimoghaddam, M., Karbasi, A.: Data summarization at scale: a two-stage submodular approach. In: ICML, pp. 3593–3602 (2018)
10. Nemhauser, G.L., Wolsey, L.A., Fisher, M.L.: An analysis of approximations for maximizing submodular set functions-i. *Math. Program.* **14**(1), 265–294 (1978)
11. Schulz, A.S., Uhan, N.A.: Approximating the least core value and least core of cooperative games with supermodular costs. *Discrete Optim.* **10**(2), 163–180 (2013)
12. Stan, S., Zadimoghaddam, M., Krause, A., Karbasi, A.: Probabilistic submodular maximization in sub-linear time. In: ICML, pp. 3241–3250 (2017)
13. Yang, R., Gu, S., Gao, C., Wu, W., Wang, H., Xu, D.: A constrained two-stage submodular maximization. *Theor. Comput. Sci.* **853**, 57–64 (2021)



# Fault-Tolerant Total Domination via Submodular Function Approximation

Ioannis Lamprou, Ioannis Sigalas, Ioannis Vaxevanakis<sup>(✉)</sup>,  
and Vassilis Zissimopoulos

Department of Informatics and Telecommunications, National and Kapodistrian  
University of Athens, Athens, Greece  
vaxjohn@di.uoa.gr

**Abstract.** In *total domination*, given a graph  $G = (V, E)$ , we seek a minimum-size set of nodes  $S \subseteq V$ , such that every node in  $V \setminus S$  has at least one neighbor in  $S$  and every node in  $S$  also has at least one neighbor in  $S$ . We define the *fault-tolerant* version of total domination, where we extend the requirement for nodes in  $V \setminus S$ . Any node in  $V \setminus S$  must have at least  $m$  neighbors in  $S$ . Let  $\Delta$  denote the maximum degree in  $G$ . We prove a first  $1 + \ln(\Delta + m - 1)$  approximation for fault-tolerant total domination. To prove our result, we develop a general submodular function approximation framework we believe is of independent interest.

**Keywords:** Fault-tolerant · Total domination · Dominating set ·  
Approximation algorithms · Submodular function

## 1 Introduction

*Domination* is a classic graph-theoretic notion which has historically attracted much attention [6, 7] in terms of combinatorial bounds, algorithmic complexity, and definition variants. In an undirected graph, a subset of its nodes is called a *dominating set* if every node of the graph is either a member of the subset or adjacent to at least one node in the subset. A *connected dominating set* is a subset of nodes such that its induced subgraph is connected and it is a dominating set.

A variant between (standard) domination and connected domination is *total domination*. A dominating set is called total if its induced subgraph does not

---

This work was partially supported by the Special Account for Research Grants (ELKE) of the National and Kapodistrian University of Athens (NKUA).

I. Lamprou—This research is co-financed by Greece and the European Union (European Social Fund - ESF) through the Operational Programme “Human Resources Development, Education and Lifelong Learning” in the context of the project “Reinforcement of Postdoctoral Researchers - 2nd Cycle” (MIS-5033021), implemented by the State Scholarships Foundation (IKY).

I. Vaxevanakis— The research work was supported by the Hellenic Foundation for Research and Innovation (HFRI) under the 3rd Call for HFRI PhD Fellowships (Fellowship Number: 5245).

include any isolated nodes. That is, both nodes outside and inside the set must be adjacent to at least one node in the set. In other words, every node must have at least one neighbor within the set. Total domination was introduced by Cockayne, Dawes, and Hedetniemi, in [2], as a natural variant of domination, and has been extensively studied since. A survey of results is provided in [8].

In this paper, we introduce *fault-tolerant* total domination. Still, every node in the graph needs to be adjacent to at least one member of the set. Nonetheless, for all nodes which are not members of the set, we require they are adjacent to at least  $m$  nodes in it, where  $m$  is part of the input. Herein, we develop a general submodular function approximation framework. We apply it to obtain a first logarithmic approximation for finding a fault-tolerant total dominating set of minimum size in general graphs. Fault-tolerance guarantees continued service in case of node failures, as serviced nodes are multiply covered. Example modern applications may be found in wireless (sensor) networking [9].

*Related Work.* The complexity of dominating set problems has been studied a lot in literature and approximation algorithms have been found. The Minimum Dominating Set (DS), Minimum Total Dominating Set (TDS) and Minimum Connected Dominating Set (CDS) problems are all NP-hard and there is no polynomial time algorithm with approximation ratio  $(1 - \varepsilon) \ln |V|$ , for any  $\varepsilon > 0$ , unless  $NP \subseteq DTIME(|V|^{O(\log \log |V|)})$  [1].

A set  $S$  is a *fault-tolerant dominating set* if every node in  $V \setminus S$  has at least  $m$  neighbors in  $S$ . In [4], a greedy algorithm with a submodular function is used to approximate fault-tolerant DS. Instead of a typical finite sums approach, like the one we use in this paper, the author employs an estimation formula to achieve a  $1 + \ln(\Delta + m)$  guarantee.

A relevant variant, with the same approximation hardness as DS, is  $k$ -tuple domination, where  $S$  is a  $k$ -tuple dominating set if every node in  $V \setminus S$  has at least  $k$  neighbors in  $S$  and every node in  $S$  has at least  $k - 1$  neighbors in  $S$ . The problem is introduced in [10], where they apply a reduction to Minimum  $k$ -Cover, a budget variant of Set Cover, to obtain a  $1 + \ln(\Delta + 1)$  guarantee.

A greedy approach for TDS in [14] yields a  $1.5 + \ln(\Delta - 0.5)$  approximation by using as potential function the sum of a submodular and a non-submodular part. Later, in Sect. 4.1, we further discuss this result in comparison to the methodology we follow in this paper. A better result is given for TDS in [1]: They reduce the problem to Set Cover and give an  $H(\Delta) - 0.5$  approximation, where  $H$  is the harmonic function. Note it holds  $\ln(n) + \frac{1}{2n} + \gamma > H(n) > \ln(n) + \gamma$ , where  $\gamma = 0.5772156649$  is the Euler constant.

Similarly to TDS, in [1], they prove a  $2 + H(\Delta)$  approximation for CDS. A better approximation for CDS is given in [11]. They use a greedy algorithm with a non-submodular function and prove a  $2 + \ln(\Delta)$  approximation. The best approximation for CDS is given in [3], where they present a  $(1 + \varepsilon)(1 + \ln(\Delta - 1))$  guarantee. For fault-tolerant CDS, in [12], they give a  $2H(\Delta + m - 1)$  approximation algorithm. Later, in [13] they improve this result to  $2 + \ln(\Delta + m - 2)$ . To achieve that, they use the same potential function as in [11], ensuring the connectivity, and add an additional function to count the extra neighbors.

*Our Results.* We develop a general framework to approximate any minimization problem able to be expressed by a submodular potential function. In Theorem 1, we give a logarithmic approximation as a function of the max and min differences when inserting a new element into a set we greedily construct. As a warm up, we apply the framework to (standard) total domination to obtain a  $1 + \ln(\Delta)$  approximation, in Theorem 2. We then arrive to fault-tolerant total domination and obtain a first  $1 + \ln(\Delta + m - 1)$  approximation in Theorem 3.

*Outline.* In Sect. 2, we define preliminary notions. In Sect. 3, we describe the general framework for submodular functions. In Sect. 4, we present our result for fault-tolerant total domination. In Sect. 5, we make concluding remarks.

## 2 Preliminaries

A graph is a pair of sets  $V, E$  with the property  $[e \in E \iff (\exists v, u \in V) e = \{v, u\}]$  and is denoted by  $G = (V, E)$ . The elements of  $V$  are called nodes and the elements of  $E$  are called edges. We assume all graphs considered in this paper are simple and connected. A node  $v$  has as neighbor a node  $u$  if there exists an edge between them ( $\{v, u\} \in E$ ). The *open neighborhood* of node  $v$  is the set of all its neighbors and is denoted by  $N(v)$ . Let the maximum degree of the graph be denoted by  $\Delta = \max_{v \in V} |N(v)|$ . The open neighborhood of  $v$  in the subset of nodes  $C \subseteq V$  is denoted by  $N_C(v) = N(v) \cap C$ . For a singleton set  $\{x\}$ , we simplify the notation from  $N_{\{x\}}(v)$  to  $N_x(v)$ .

Let  $U$  denote a universe of elements. The set of all subsets of  $U$  is denoted by  $2^U$ . A function  $f : 2^U \rightarrow \mathbb{R}$  is called *monotone increasing* if for any  $A \subseteq B \subseteq U$  it holds  $f(A) \leq f(B)$ . Let  $\Delta_x f(A) = f(A \cup \{x\}) - f(A)$ . A function  $f : 2^U \rightarrow \mathbb{R}$  is *submodular* if for any  $A \subseteq B \subseteq U, x \in U$ , it holds  $\Delta_x f(B) \leq \Delta_x f(A)$ .

A set  $S \subseteq V$  is a *dominating set* of  $G$  if every node in  $V \setminus S$  has at least one neighbor in  $S$ .  $S$  is a *fault-tolerant dominating set* if every node in  $V \setminus S$  has at least  $m$  neighbors in  $S$ .  $S$  is a *total dominating set* if  $S$  is a dominating set and every node in  $S$  has at least one neighbor in  $S$ . Equivalently, every node in  $V$  has at least one neighbor in  $S$ .  $S$  is a *fault-tolerant total dominating set* if it is fault-tolerant dominating and every node in  $S$  has at least one neighbor in  $S$ .

In this paper, we examine the following optimization problems.

**Definition 1 (Minimum Total Dominating Set).** *Given a graph  $G = (V, E)$  find a total dominating set  $S \subseteq V$  of minimum cardinality.*

**Definition 2 (Minimum Fault-Tolerant Total Dominating Set).** *Given a graph  $G = (V, E)$ , find a fault-tolerant total dominating set  $S \subseteq V$  of minimum cardinality.*

## 3 General Greedy Submodular Approximation

In this section, we present a general greedy approximation algorithm which can be applied to minimization problems described by a submodular and monotone

increasing function. Intuitively, the algorithm constructs a subset by iteratively picking the element whose insertion maximizes the function at that point. The algorithm is general in the sense that it works for any given set of *seeds*, that is, elements required to be in the final solution.

Let  $U$  be a finite set (universe) of elements and  $X \subseteq U$  be a seed set given as input. For our purposes, a problem with universe  $U$  and seed set  $X$  may be defined as a function  $Q_X : 2^U \rightarrow \{0, 1\}$ , such that for any  $S \in 2^U$ , it holds  $Q_X(S) = 1$  if and only if the set  $X \cup S$  is a feasible solution to the problem. Let  $\mathcal{Q}_X = \{S \subseteq U : Q_X(S) = 1\}$ . Assume there is a submodular monotone increasing function  $f : 2^U \rightarrow \mathbb{R}$  with the property, for any  $C \subseteq U$  where  $X \cap C = \emptyset$ , it holds  $f(X \cup C) = \max_{U' \subseteq U} f(U')$  if and only if  $C \in \mathcal{Q}_X$ . Let  $S^*$  be a member of  $\mathcal{Q}_X$  with minimum cardinality, where  $X \cap S^* = \emptyset$ . In Algorithm 1, we introduce Greedy Constructor to return a solution approximating the size of  $S^*$ .

---

**Algorithm 1:** Greedy Constructor

---

**Input:** Universe  $U$ , Seed Set  $X$   
**Output:**  $S \in \mathcal{Q}_X$

- 1  $S \leftarrow \emptyset$
- 2 **while**  $(\exists u \in U \setminus (X \cup S)) \Delta_u f(X \cup S) > 0$  **do**
- 3      $x \leftarrow \operatorname{argmax}_{u \in U \setminus (X \cup S)} \Delta_u f(X \cup S)$
- 4      $S \leftarrow S \cup \{x\}$
- 5 **end while**
- 6 Return  $S$

---

To begin with the analysis of Greedy Constructor, we first define what is a *Greedy Maximum Differential Set*, which will be returned by the algorithm.

**Definition 3 (Greedy Maximum Differential Set).** *Let  $U \subseteq \mathbb{N}$ , a seed set  $X \subseteq U$  and  $f : 2^U \rightarrow \mathbb{R}$ . Let  $S \subseteq U$  and a total order of the elements in  $S$ , namely  $s_1, s_2, \dots$ . Let  $S_i = \{s_1, s_2, \dots, s_i\} \subseteq S$ , where  $S_0 = \emptyset$ . A set  $S \subseteq U$  is called a Greedy Maximum Differential Set of  $f$  generated by  $X$  if there is a total order of the elements of  $S$  such that for each  $s_i$  it holds  $\Delta_{s_i} f(X \cup S_{i-1}) \geq \Delta_u f(X \cup S_{i-1})$  for all  $u \in U$ .*

**Definition 4.** *The set of all Greedy Maximum Differential Sets of  $f$  generated by  $X$  is denoted by  $D_f(X)$ . In case  $X = \emptyset$ , it is simplified the notation to  $D_f$ .*

Let  $f$  be a submodular and monotone increasing function and  $S \in D_f(X)$ . By submodularity of  $f$ , we denote  $\Delta_{max} := \max_{s_i \in S} \Delta_{s_i} f(X \cup S_{i-1}) = \Delta_{s_1} f(X)$  and  $\Delta_{min} := \min_{s_i \in S} \Delta_{s_i} f(X \cup S_{i-1}) = \Delta_{s_{|S|}} f(X \cup S_{|S|-1})$  as the maximum and minimum differences, respectively, when inserting a new element into  $S$ . Since  $f$  is monotone increasing, we denote  $f_{max} := \max_{C \subseteq U} f(C) = f(U)$ . In the following key lemma, we bound the size of a greedy maximum differential set  $S$  to be a logarithmic approximation of the size of an optimal solution achieving  $f_{max}$ .



**Lemma 1.** Let  $U$  be a finite set,  $f : 2^U \rightarrow \mathbb{R}$  submodular and monotone increasing function and  $X \subseteq U$ . Let  $S \subseteq U$  with the properties:

- $S \in D_f(X)$
- $f(X \cup S) = f_{max}$
- $\Delta_{min} > 0$ .

For every set  $C \subseteq U$  with  $X \cap C = \emptyset$  and  $f(X \cup C) = f_{max}$ , it holds

$$|S| \leq \left( 1 + \ln \left( \frac{\Delta_{max}}{\Delta_{min}} \right) \right) \cdot |C|$$

*Proof.* Let  $S = \{s_1, s_2, \dots, s_{|S|}\} \in D_f(X)$  such that  $f(X \cup S) = f_{max}$  and  $\Delta_{min} > 0$ . Let  $C = \{c_1, c_2, \dots, c_{|C|}\} \subseteq U$  with  $f(X \cup C) = f_{max}$ , where  $c_1, c_2, \dots, c_{|C|}$  is an arbitrary order of the elements of  $C$ . Let  $S_i = \{s_1, s_2, \dots, s_i\}$ , and  $C_i = \{c_1, c_2, \dots, c_i\}$ , where  $S_0 = C_0 = \emptyset$ .

Since  $f(X \cup C) = f_{max}$  and  $f$  is monotone increasing, we get  $f(X \cup S_i \cup C) = f_{max}$  for any set  $S_i$ . It follows,

$$\begin{aligned} f_{max} - f(X \cup S_{i-1}) &= f(X \cup S_{i-1} \cup C) - f(X \cup S_{i-1}) \\ &= f(X \cup S_{i-1} \cup C_{|C|}) - f(X \cup S_{i-1} \cup C_0) \\ &= f(X \cup S_{i-1} \cup C_{|C|}) \\ &\quad - f(X \cup S_{i-1} \cup C_{|C|-1}) + f(X \cup S_{i-1} \cup C_{|C|-1}) \\ &\quad \vdots \\ &\quad - f(X \cup S_{i-1} \cup C_1) + f(X \cup S_{i-1} \cup C_1) \\ &\quad - f(X \cup S_{i-1} \cup C_0) \\ &= \sum_{j=1, \dots, |C|} f(X \cup S_{i-1} \cup C_j) - f(X \cup S_{i-1} \cup C_{j-1}) \\ &= \sum_{j=1, \dots, |C|} \Delta_{c_j} f(X \cup S_{i-1} \cup C_{j-1}) \end{aligned}$$

since by definition  $C_{|C|} = C$ ,  $C_0 = \emptyset$ .

Since  $f$  is submodular, for any  $c_j$ , it follows  $\Delta_{c_j} f(X \cup S_{i-1} \cup C_{j-1}) \leq \Delta_{c_j} f(X \cup S_{i-1})$ . Let  $c_{j'}$   $\in C$  be the element maximizing  $\Delta_{c_j} f(X \cup S_{i-1})$ . Then,

$$\begin{aligned} \sum_{j=1, \dots, |C|} \Delta_{c_j} f(X \cup S_{i-1}) &\leq |C| \cdot \Delta_{c_{j'}} f(X \cup S_{i-1}) \\ &\leq |C| \cdot \Delta_{s_i} f(X \cup S_{i-1}) \end{aligned}$$

where  $\Delta_{c_{j'}} f(X \cup S_{i-1}) \leq \Delta_{s_i} f(X \cup S_{i-1})$ , since  $S \in D_f$  (Definition 3).

Overall, we have

$$\begin{aligned}
 f_{max} - f(X \cup S_{i-1}) &\leq |C| \cdot \Delta_{s_i} f(X \cup S_{i-1}) \\
 \frac{f_{max} - f(X \cup S_{i-1})}{|C|} &\leq f(X \cup S_i) - f(X \cup S_{i-1}) \\
 -f(X \cup S_i) &\leq -f(X \cup S_{i-1}) - \frac{f_{max} - f(X \cup S_{i-1})}{|C|} \\
 f_{max} - f(X \cup S_i) &\leq f_{max} - f(X \cup S_{i-1}) - \frac{f_{max} - f(X \cup S_{i-1})}{|C|}
 \end{aligned} \tag{1}$$

Let  $a_i = f_{max} - f(X \cup S_i)$  for any  $i$ . Then, by induction it follows

$$a_i \leq a_{i-1} - \frac{a_{i-1}}{|C|} = a_{i-1} \left(1 - \frac{1}{|C|}\right) \leq \dots \leq a_0 \left(1 - \frac{1}{|C|}\right)^i \leq a_0 \cdot e^{-\frac{i}{|C|}}$$

since for any  $x \in \mathbb{R}$  it holds  $(1 + x) \leq e^x$ .

**Proposition 1.** *Let  $\delta > 0$ . For every  $k \in \{0, \dots, |S|\}$ , if  $a_k \leq \delta \cdot |C|$ , then  $|S| \leq \frac{\delta \cdot |C|}{\Delta_{min}} + k$ . Also, if  $a_k < \delta \cdot |C|$ , then  $|S| < \frac{\delta \cdot |C|}{\Delta_{min}} + k$ .*

We continue the proof the lemma. For some  $\delta > 0$ , we distinguish two cases.

If  $a_0 \leq \delta \cdot |C|$ , then by Proposition 1 it holds  $|S| \leq \frac{\delta \cdot |C|}{\Delta_{min}}$  (case I).

If  $a_0 > \delta \cdot |C|$ , then since  $a_i$  is monotonically decreasing, by definition there exists  $i_0$  such that

$$a_{i_0+1} < \delta \cdot |C| \leq a_{i_0}.$$

We first upper bound  $i_0$  based on the right inequality. Recall that  $a_{i_0} \leq a_0 \cdot e^{-\frac{i_0}{|C|}}$ .

$$\begin{aligned}
 \delta \cdot |C| &\leq a_0 \cdot e^{-\frac{i_0}{|C|}} \\
 e^{\frac{i_0}{|C|}} &\leq \frac{a_0}{\delta \cdot |C|} \\
 \frac{i_0}{|C|} &\leq \ln\left(\frac{a_0}{\delta \cdot |C|}\right) = \ln\left(\frac{1}{\delta}\right) + \ln\left(\frac{a_0}{|C|}\right) \\
 i_0 &\leq \left(\ln\left(\frac{a_0}{|C|}\right) - \ln(\delta)\right) \cdot |C| \stackrel{(1)}{\leq} \left(\ln\left(\Delta_{s_1} f(X \cup S_0)\right) - \ln(\delta)\right) \cdot |C| \\
 i_0 &\leq \left(\ln\left(\Delta_{s_1} f(X)\right) - \ln(\delta)\right) \cdot |C| \\
 i_0 &\leq \left(\ln\left(\Delta_{max}\right) - \ln(\delta)\right) \cdot |C|
 \end{aligned} \tag{2}$$

To complete the proof, we now upper bound  $S$  based on left inequality. Since  $a_{i_0+1} < \delta \cdot |C|$ , by Proposition 1, it follows  $|S| < \frac{\delta \cdot |C|}{\Delta_{min}} + i_0 + 1$  (case II).

Note that it suffices to examine case II, since it contains case I. It follows:

$$|S| < \frac{\delta \cdot |C|}{\Delta_{min}} + i_0 + 1 \leq \frac{\delta \cdot |C|}{\Delta_{min}} + (\ln(\Delta_{max}) - \ln(\delta)) \cdot |C| + 1$$

$$|S| < \left( \frac{\delta}{\Delta_{min}} - \ln(\delta) + \ln(\Delta_{max}) \right) \cdot |C| + 1$$

Now we find a value of  $\delta$  which minimizes the upper bound of  $S$ . Let  $g(\delta) = \frac{\delta}{\Delta_{min}} - \ln(\delta) + \ln(\Delta_{max})$ . The function  $g$  is minimized when  $\delta = \Delta_{min}$  because:

$$g'(\delta) = \frac{1}{\Delta_{min}} - \frac{1}{\delta} \quad \text{and} \quad g'(\delta) = 0 \iff \delta = \Delta_{min}$$

Recall  $|S| < \frac{\delta \cdot |C|}{\Delta_{min}} + i_0 + 1$ . Since we choose  $\delta = \Delta_{min}$ , it follows  $|S| < |C| + i_0 + 1$ , and so:

$$|S| \leq |C| + i_0 \stackrel{(2)}{\leq} |C| + \left( \ln(\Delta_{max}) - \ln(\Delta_{min}) \right) \cdot |C| \leq \left( 1 + \ln \left( \frac{\Delta_{max}}{\Delta_{min}} \right) \right) \cdot |C|.$$

**Lemma 2.** *Let  $S$  be the set returned by Algorithm 1. It holds  $f(X \cup S) = f_{max}$ .*

*Proof.* Let  $f(X \cup S) < f_{max}$ . We distinguish two cases:

case 1: Let  $X \cup S = U$ . Then  $f(X \cup S) = f(U) = f_{max}$  and it is a contradiction.

case 2: Let  $X \cup S \subset U$  and  $U^c = U \setminus (X \cup S)$ . By definition of Algorithm 1, it holds  $\Delta_s f(X \cup S) = 0$  for every  $s \in U^c$ , otherwise, the while loop of Algorithm 1 would not stop and so  $X \cup S = U$ . Since  $f$  is submodular and monotone increasing, for every  $S' \supset X \cup S$  and  $s' \in U \setminus S'$  it holds  $\Delta_{s'} f(S') \leq \Delta_{s'} f(X \cup S) = 0$  and so  $\Delta_{s'} f(S') = 0$ . We have  $f(U) - f(X \cup S) = 0$  because:

$$\begin{aligned} f(U) - f(X \cup S) &= f\left(X \cup S \cup \bigcup_{i=1}^{|U^c|} \{s'_i\}\right) && - f(X \cup S) \\ &= f\left(X \cup S \cup \bigcup_{i=1}^{|U^c|} \{s'_i\}\right) && - f\left(X \cup S \cup \bigcup_{i=1}^{|U^c|-1} \{s'_i\}\right) \\ &\quad + f\left(X \cup S \cup \bigcup_{i=1}^{|U^c|-1} \{s'_i\}\right) && - f\left(X \cup S \cup \bigcup_{i=1}^{|U^c|-2} \{s'_i\}\right) \\ &\quad \vdots \\ &\quad + f\left(X \cup S \cup \{s'_1, s'_2\}\right) && - f\left(X \cup S \cup \{s'_1\}\right) \\ &\quad + f\left(X \cup S \cup \{s'_1\}\right) && - f(X \cup S) \\ &= \sum_{j=1}^{|U^c|} \Delta_{s'_{|U^c|+1-j}} f\left(X \cup S \cup \bigcup_{i=1}^{|U^c|-j} \{s'_i\}\right) = 0 \end{aligned}$$

Thus,  $f(X \cup S) = f(U) = f_{max}$ , a contradiction.

**Theorem 1.** *Algorithm 1 returns a  $\left(1 + \ln\left(\frac{\Delta_{max}}{\Delta_{min}}\right)\right)$ -approximation to the minimum size set in  $\mathcal{Q}_{\mathcal{X}}$ .*

## 4 Fault-Tolerant Total Domination

In this section, we present an application of the general approximation framework given in Sect. 3. We show how to apply the framework (with an empty seed set) to obtain a first logarithmic approximation for fault-tolerant total domination. As a warm up, we first consider the case of (standard) total domination.

### 4.1 Total Domination

Recall that given a graph  $G = (V, E)$ , a subset of nodes  $S \subseteq V$  is a total dominating set if every node outside of  $S$ , that is  $v \in V \setminus S$ , has at least one neighbor in  $S$  and  $S$  has no isolated nodes. Equivalently, every node in  $V$  has a neighbor in  $S$ . We define a submodular and monotone increasing function  $f$  such that any subset of  $V$  achieving  $f_{max}$  is a total dominating set.

**Definition 5.** *Let  $G = (V, E)$  be a graph. We define  $f : 2^V \rightarrow \mathbb{R}$  as follows*

$$f(A) = \sum_{v \in V} \delta_A(v)$$

where

$$\delta_A(v) = \begin{cases} 1, & |N_A(v)| > 0 \\ 0, & \text{otherwise.} \end{cases}$$

Intuitively,  $f(A)$  is the number of nodes having at least one neighbor in  $A$ .

**Definition 6.** *For  $G = (V, E)$  and any  $A \subseteq V$ , let  $K(A) = \{v \in V : \delta_A(v) = 1\}$  be the set of nodes that have at least one neighbor in  $A$ .*

**Lemma 3.** *Let  $G = (V, E)$  be a graph,  $A \subseteq V$  and  $x \in V \setminus A$ . Then:*

$$f(A \cup \{x\}) = f(A) + |N_{V \setminus K(A)}(x)|$$

*Proof.* By definition of  $K$  and  $\delta$ , for every  $v \in K(A)$  it holds  $\delta_{A \cup \{x\}}(v) = \delta_A(v) = 1$  and for every  $v \notin K(A)$  it holds  $\delta_{A \cup \{x\}}(v) = \delta_A(v) + |N_x(v)| = |N_x(v)|$ . We have  $f(A \cup \{x\}) = \sum_{v \in V} \delta_{A \cup \{x\}}(v)$ , which we show

$$\begin{aligned} \sum_{v \in V} \delta_{A \cup \{x\}}(v) &= \sum_{v \in K(A)} \delta_{A \cup \{x\}}(v) + \sum_{v \notin K(A)} \delta_{A \cup \{x\}}(v) \\ &= \sum_{v \in K(A)} \delta_A(v) + \sum_{v \notin K(A)} (\delta_A(v) + |N_x(v)|) \\ &= \sum_{v \in K(A)} \delta_A(v) + \sum_{v \notin K(A)} \delta_A(v) + \sum_{v \notin K(A)} |N_x(v)| \\ &= \sum_{v \in V} \delta_A(v) + \sum_{v \notin K(A)} |N_x(v)| = f(A) + |N_{V \setminus K(A)}(x)|. \end{aligned}$$

**Lemma 4.** *Function  $f$  (Definition 5) is submodular and monotone increasing.*

*Proof.* First, we show  $f$  is monotone increasing. For every  $A, B$  such that  $A \subseteq B \subseteq V$  and for every  $v \in V$ , by applying definitions, it holds  $N_A(v) \subseteq N_B(v)$ , which implies  $\delta_A(v) \leq \delta_B(v)$  and so  $f(A) \leq f(B)$ .

Second, we show  $f$  is submodular. For every  $A, B$  such that  $A \subseteq B \subseteq V$  and for every  $x \in V \setminus B$ , by definition of  $K$ , it holds  $K(A) \subseteq K(B) \Rightarrow V \setminus K(B) \subseteq V \setminus K(A)$  and so  $|N_{V \setminus K(B)}(x)| \leq |N_{V \setminus K(A)}(x)|$ , which implies  $f(B) + |N_{V \setminus K(B)}(x)| - f(B) \leq f(A) + |N_{V \setminus K(A)}(x)| - f(A)$ . By Lemma 3, it holds  $f(A \cup \{x\}) = f(A) + |N_{V \setminus K(A)}(x)|$  and  $f(B \cup \{x\}) = f(B) + |N_{V \setminus K(B)}(x)|$  and so  $f(B \cup \{x\}) - f(B) \leq f(A \cup \{x\}) - f(A)$ .

**Lemma 5.** *Let  $G = (V, E)$  be a graph. A set  $S \subseteq V$  is a total dominating set if and only if  $f(S) = f_{max}$ .*

*Proof.* Note  $f_{max} = f(V) = |V|$  since for all  $v \in V$  it holds  $|N_V(v)| > 0$  and so  $\delta_V(v) = 1$ .

Let  $S \subseteq V$  be a total dominating set. Since  $S$  is total dominating set, then  $\delta_S(v) = 1$  for all  $v \in V$ . So, we get  $f(S) = \sum_{v \in V} \delta_S(v) = |V| = f_{max}$ .

Consider the case  $S \subseteq V$  is not a total dominating set and  $f(S) = f_{max} = |V|$ . Then, there exists  $w \in V$  such that  $N_S(w) = 0 \Rightarrow \delta_S(w) = 0$ . So,  $f(S) = \sum_{v \in V} \delta_S(v) = \sum_{v \in V \setminus \{w\}} \delta_S(v) < |V| = f_{max}$ , a contradiction.

**Theorem 2.** *Algorithm 1, where  $X = \emptyset, U = V$ , returns a  $(1 + \ln(\Delta))$ -approximation for Minimum Total Dominating Set.*

*Discussion.* Let us briefly comment on why the analysis performed in [14] falls short of our approximation guarantee. Below, let  $f$  be our potential function (Definition 5),  $f'$  be the potential function defined in [14] and  $T^* = \{y_1, \dots, y_{|T^*|}\}$  be the minimum total dominating set for a given input. In [14],  $f'$  comprises two parts  $f'(T) = i(T) + w(T)$ , where  $i(T)$  denotes the number of nodes in  $T$  which are not adjacent to  $T$ , that is, the number of isolated nodes within  $T$ , and  $w(T)$  is the number of nodes outside  $T$  which are not adjacent to  $T$ . While they prove  $w$  is submodular, they do not prove the same for  $i$ . To overcome this obstacle, they observe  $\Delta_{y_j} i(T_{i-1} \cup T_{j-1}^*) \leq \Delta_{y_j} i(T_{i-1}) + m_j$ , where  $m_j = 1$  if  $y_j$  is not adjacent to  $T_{j-1}^*$  and  $m_j = 0$  otherwise. Then, it follows  $a_i \leq a_{i-1} - \frac{a_{i-1}}{|T^*|} + \frac{m}{|T^*|}$ , where  $m = \sum_{j=1}^{|V|} m_j$ . Since  $T^*$  is a total dominating set, they observe  $m \leq \frac{|T^*|}{2}$ , which leads to  $a_i \leq a_{i-1} - \frac{a_{i-1}}{|T^*|} + \frac{1}{2}$ . Instead, in our analysis, we prove  $f$  is submodular and as a result we arrive to the inequality  $a_i \leq a_{i-1} - \frac{a_{i-1}}{|T^*|}$ . Overall, we improve from  $1.5 + \ln(\Delta - 0.5)$  to  $1 + \ln(\Delta)$ .

### 4.2 Fault-Tolerant Total Domination

In this subsection, we generalize to the fault-tolerant case of total domination. Recall that given a graph  $G = (V, E)$ , a subset of nodes  $S \subseteq V$  is a fault-tolerant

total dominating set if every node outside of  $S$ , that is,  $v \in V \setminus S$ , has at least  $m$  neighbors in  $S$  and  $S$  has no isolated nodes. We define a submodular and monotone increasing function  $f$  such that any subset of  $V$  achieving  $f_{max}$  is a fault-tolerant total dominating set.

**Definition 7.** Let  $G = (V, E)$  be a graph. We define  $f : 2^V \rightarrow \mathbb{R}$  as follows:

$$f(A) = \sum_{v \in V} m_A(v)$$

where:

$$m_A(v) = \begin{cases} m, & [v \notin A \wedge |N_A(v)| \geq m] \vee [v \in A \wedge |N_A(v)| > 0] & (a) \\ m - 1, & v \in A \wedge |N_A(v)| = 0 & (b) \\ |N_A(v)|, & \text{otherwise.} & (c) \end{cases}$$

Intuitively,  $m_A(v)$  is the potential of node  $v$  toward satisfying the problem definition. When  $v$  fully meets the definition requirements for total domination, it is assigned a value of  $m$  (case a). If  $v \notin A$  and has  $m' < m$  neighbors in  $A$ , then we assign it a value of  $m'$  (case c). If  $v \in A$ , then it meets the definition when it has at least one neighbor in  $A$ . In case it does not have a neighbor in  $A$ , we artificially assign it a value of  $m - 1$  (case b). The value will increase to  $m$  only when there appears a neighbor of  $v$  in  $A$ .

**Definition 8.** For  $G = (V, E)$  and  $A \subseteq V$ , let  $K(A) = \{v \in V : m_A(v) = m\}$  be the set of nodes that have at least one neighbor in  $A$ , if the node is in  $A$ , or have at least  $m$  neighbors in  $A$ , if the node is in  $V \setminus A$ .

**Lemma 6.** Let  $G = (V, E)$ ,  $A \subseteq V$  and  $x \in V \setminus A$ . Then:

$$f(A \cup \{x\}) = f(A) + |N_{V \setminus K(A)}(x)| + t_A(x)$$

where

$$t_A(x) = \begin{cases} 0, & x \in K(A) \\ m - |N_A(x)|, & x \notin K(A) \wedge |N_A(x)| > 0 \\ m - 1, & x \notin K(A) \wedge |N_A(x)| = 0 \end{cases}$$

Before proceeding to the proof, we provide some intuition on the definition of  $t_A(x)$ . Overall, it holds  $t_A(x) = m_{A \cup \{x\}}(x) - m_A(x)$ , that is,  $t_A(x)$  captures the increase in the potential of  $x$  attributed to the insertion of  $x$  into  $A$ .

*Proof (Proof of Lemma 6).* We first compute an expression for the value of  $m_{A \cup \{x\}}(v)$  for any  $v \in V$ , where  $v \neq x$ . If  $v \in K(A)$ , then it holds  $m_{A \cup \{x\}}(v) = m_A(v) = m$ . Otherwise, if  $v \notin K(A)$ , let us show  $m_{A \cup \{x\}}(v) = m_A(v) + |N_x(v)|$ .

- If  $v \in A$ , then  $|N_A(v)| = 0$  by definition of  $K(A)$ , so we are in case (b) of Definition 7 and  $m_A(v) = m - 1$ . Since  $v \in A$ , it also holds  $v \in A \cup \{x\}$ .

- If  $(v, x) \in E$ , then  $|N_{A \cup \{x\}}(v)| > 0$ , and by case (a) of Definition 7, it follows  $m_{A \cup \{x\}}(v) = m = m - 1 + 1 = m_A(v) + |N_x(v)|$ .
  - If  $(v, x) \notin E$ , then  $|N_{A \cup \{x\}}(v)| = |N_A(v)| + |N_x(v)| = 0 + 0 = 0$ , so  $m_{A \cup \{x\}}(v) = m - 1 = m - 1 + 0 = m_A(v) + |N_x(v)|$ .
- If  $v \notin A$ , then  $v \notin A \cup \{x\}$ . We are in case (c) of Definition 7, so it holds  $m_A(v) = |N_A(v)|$ . Since  $v \notin A \cup \{x\}$ , we compute  $m_{A \cup \{x\}}(v) = |N_{A \cup \{x\}}(v)| = \sum_{u \in A \cup \{x\}} |N_u(v)| = \sum_{u \in A} |N_u(v)| + |N_x(v)| = |N_A(v)| + |N_x(v)| = m_A(v) + |N_x(v)|$ .

For the new node  $x$ , it holds  $m_{A \cup \{x\}}(x) = m_A(x) + t_A(x)$ , since

- If  $x \in K(A)$ , then  $m_{A \cup \{x\}}(x) = m_A(x) = m$ .
- If  $x \notin K(A)$  and  $|N_A(x)| > 0$ , then  $m_{A \cup \{x\}}(x) = m = |N_A(x)| + m - |N_A(x)| = m_A(x) + m - |N_A(x)|$ .
- If  $x \notin K(A)$  and  $|N_A(x)| = 0$ , then  $m_{A \cup \{x\}}(x) = m - 1 = 0 + m - 1 = m_A(x) + m - 1$ .

We now compute the value of  $f(A \cup \{x\})$

$$\begin{aligned}
 f(A \cup \{x\}) &= \sum_{v \in V} m_{A \cup \{x\}}(v) = \\
 &\sum_{\substack{v \in K(A) \\ v \neq x}} m_{A \cup \{x\}}(v) + \sum_{\substack{v \notin K(A) \\ v \neq x}} m_{A \cup \{x\}}(v) + m_{A \cup \{x\}}(x) = \\
 &\sum_{\substack{v \in K(A) \\ v \neq x}} m_A(v) + \sum_{\substack{v \notin K(A) \\ v \neq x}} (m_A(v) + |N_x(v)|) + m_A(x) + m_{A \cup \{x\}}(x) - m_A(x) = \\
 &\sum_{\substack{v \in K(A) \\ v \neq x}} m_A(v) + \sum_{\substack{v \notin K(A) \\ v \neq x}} m_A(v) + m_A(x) + \sum_{\substack{v \notin K(A) \\ v \neq x}} |N_x(v)| + t_A(x) = \\
 &\sum_{v \in V} m_A(v) + \sum_{v \notin K(A)} |N_x(v)| + t_A(x) = f(A) + |N_{V \setminus K(A)}(x)| + t_A(x).
 \end{aligned}$$

**Lemma 7.** *Function  $f$  is submodular and monotone increasing.*

**Lemma 8.** *Let  $G = (V, E)$  be a graph. A set  $S \subseteq V$  is a fault-tolerant total dominating set if and only if  $f(S) = f_{max}$ .*

*Proof.* We get  $f_{max} = f(V) = m|V|$ , since, for all  $v \in V$ , it holds  $|N_V(v)| > 0$  and so  $m_V(v) = m$ .

Let  $S \subseteq V$  be a fault-tolerant total dominating set. Then,  $m_V(v) = m$  for all  $v \in V$ . So,  $f(S) = \sum_{v \in V} m_S(v) = m|V| = f_{max}$ .

Assume  $S \subseteq V$  is not a fault-tolerant total dominating set and  $f(S) = f_{max} = m|V|$ . Then, there exists  $v' \in V \setminus S$  such that  $|N_S(v')| < m$  or there exists  $s' \in S$  such that  $|N_S(s')| = 0$ . So, there exists  $w \in V$  such that  $m_S(w) < m$  since  $|N_S(v')| < m \Rightarrow m_S(v') < m$  and  $|N_S(s')| = 0 \Rightarrow m_S(s') = m - 1 < m$  and  $f(S) = \sum_{v \in V \setminus \{w\}} m_S(v) + m_S(w) < m(|V| - 1) + m = f_{max}$ , a contradiction.

**Theorem 3.** *Algorithm 1, where  $X = \emptyset$ ,  $U = V$ , returns a  $(1 + \ln(\Delta + m - 1))$ -approximation for Minimum Fault-Tolerant Total Dominating Set.*

## 5 Conclusions

We develop a general framework to greedily approximate a family of problems captured by a submodular potential function. We prove a first logarithmic approximation for Fault-Tolerant Total Domination by applying the framework. With minimal work, our proofs can be shown to stand for any seed set choice. In the future, we plan to apply the framework to problems involving seed sets, e.g., in biology inspired applications like disease pathway problems [5] or other.

## References

1. Chlebík, M., Chlebíková, J.: Approximation hardness of dominating set problems in bounded degree graphs. *Inf. Comput.* **206**(11), 1264–1275 (2008)
2. Cockayne, E.J., Dawes, R.M., Hedetniemi, S.T.: Total domination in graphs. *Networks* **10**(3), 211–219 (1980)
3. Du, D.Z., Graham, R.L., Pardalos, P.M., Wan, P.J., Wu, W., Zhao, W.: Analysis of greedy approximations with nonsubmodular potential functions. In: *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 167–175 (2008)
4. Foerster, K.-T.: Approximating fault-tolerant domination in general graphs. In: *2013 Proceedings of the Tenth Workshop on Analytic Algorithmics and Combinatorics (ANALCO)*, pp. 25–32. SIAM (2013)
5. Ghiassian, S.D., Menche, J., Barabási, A.-L.: A disease module detection (diamond) algorithm derived from a systematic analysis of connectivity patterns of disease proteins in the human interactome. *PLoS Comput. Biol.* **11**(4), e1004120 (2015)
6. Haynes, T.W., Hedetniemi, S., Slater, P.: *Fundamentals of Domination in Graphs*. CRC Press, Boca Raton (2013)
7. Haynes, T.W., Hedetniemi, S.T., Henning, M.A.: *Topics in Domination in Graphs*, vol. 64. Springer, Cham (2020). <https://doi.org/10.1007/978-3-030-51117-3>
8. Henning, M.A., Yeo, A.: *Total Domination in Graphs*. Springer, New York (2013). <https://doi.org/10.1007/978-1-4614-6525-6>
9. Jena, S.K., Das, G.K.: Total domination in unit disk graphs. *CoRR*, abs/2007.11997 (2020)
10. Klasing, R., Laforest, C.: Hardness results and approximation algorithms of k-tuple domination in graphs. *Inf. Process. Lett.* **89**(2), 75–83 (2004)
11. Ruan, L., Hongwei, D., Jia, X., Weili, W., Li, Y., Ker, I.K.: A greedy approximation for minimum connected dominating sets. *Theoret. Comput. Sci.* **329**(1–3), 325–330 (2004)
12. Zhang, Z., Liu, Q., Li, D.: Two algorithms for connected r-hop k-dominating set. *Discrete Math. Algorithms Appl.* **1**(04), 485–498 (2009)
13. Zhou, J., Zhang, Z., Weili, W., Xing, K.: A greedy algorithm for the fault-tolerant connected dominating set in a general graph. *J. Comb. Optim.* **28**(1), 310–319 (2014)
14. Zhu, J.: Approximation for minimum total dominating set. In: *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human*. ICIS 2009, pp. 119–124. ACM, New York (2009)





# On the Parallel Complexity of Constrained Read-Once Refutations in UTVPI Constraint Systems

K. Subramani<sup>(✉)</sup> and Piotr Wojciechowski

West Virginia University, Morgantown, WV, USA  
{k.subramani,pwojciec}@mail.wvu.edu

**Abstract.** This paper is concerned with the parallel complexity of the constraint-required read-once refutation (CROR) problem in Unit Two Variable Per Inequality (UTVPI) constraint systems. Recall that a UTVPI constraint is a linear inequality of the form:  $a_i \cdot x_i + a_j \cdot x_j \leq b_k$ , where  $a_i, a_j \in \{0, 1, -1\}$  and  $b_k \in \mathbb{Z}$ . A conjunction of such constraints is called a UTVPI constraint system (UCS) and can be represented in matrix form as:  $\mathbf{A} \cdot \mathbf{x} \leq \mathbf{b}$ . UTVPI constraints are used in many domains including operations research and program verification. A refutation is a proof of infeasibility. A read-once refutation (ROR) is a refutation in which each constraint is used at most once. We focus on a variant of the ROR problem in which we specify which constraints a refutation is required to use. This variant is known as the CROR problem. In this paper, we provide **NC** reductions between the CROR problem in UCSs and the decision version of the minimum weight perfect matching problem.

## 1 Introduction

This paper examines a problem associated with linearly infeasible systems of Unit Two Variable Per Inequality (UTVPI) constraints. A linear relationship of the form:  $a_i \cdot x_i + a_j \cdot x_j \leq b_k$  is called a UTVPI constraint, if  $a_i, a_j \in \{0, 1, -1\}$ . A conjunction of such constraints is called a UCS. Observe that a UCS is a specialized linear program (LP) and thus can be represented in matrix form as:  $\mathbf{U} : \mathbf{A} \cdot \mathbf{x} \leq \mathbf{b}$ . This means that if  $\mathbf{U}$  has no linear (rational) solutions, then there exists a non-negative vector  $\mathbf{y}$ , such that  $\mathbf{y} \cdot \mathbf{A} = \mathbf{0}$ ,  $\mathbf{y} \cdot \mathbf{b} < 0$ . This follows directly from Farkas' Lemma [6]. The vector  $\mathbf{y}$  serves as a refutation of the feasibility of  $\mathbf{U}$  in that it “proves” that  $\mathbf{U}$  has no linear solutions. A read-once refutation (ROR) is a refutation that uses each constraint at most once. Thus, a refutation  $\mathbf{y}$  is read-once, if each element  $y_i$  of  $\mathbf{y}$  belongs to the set  $\{0, 1\}$ .

Of the various forms of linear refutation, read-once refutations can be considered to be the simplest. Observe that a read-once refutation of a system  $\mathbf{U}$  corresponds to a subset of the constraints of  $\mathbf{U}$ , which when summed together causes contradiction. This is in contrast to more general forms of refutation where the number of times each constraint is used is important. Additionally,

since coefficients are unnecessary for read-once refutations, read-once refutations are more compact than more general forms of refutation. This makes read-once refutations a highly desirable proof of infeasibility.

Unfortunately, read-once refutation is an **incomplete** proof system, since there exist infeasible LPs that do not have such a refutation. In fact, this is also the case for UCSs [18]. Consequently, the problem of checking if an arbitrary UCS has a read-once refutation is interesting.

The primary focus of this paper is a variant of the ROR problem in which we are given a set of constraints that the refutation is required to use. This variant is known as the constraint-required read-once refutation (CROR) problem. This makes the problem different from, and possibly harder than, the unrestricted ROR problem. We provide **NC** reductions between the CROR problem in UCSs and the decision version of the minimum weight perfect matching (MWPM<sub>D</sub>) problem. These reductions prove that the CROR problem in UCSs is **NC**-equivalent to the MWPM<sub>D</sub> problem.

The rest of the paper is organized as follows: Sect. 2 formally describes the problems under consideration. The motivation for our work and related approaches in the literature are described in Sect. 3. In Sect. 4, we provide **NC** reductions between CROR and MWPM<sub>D</sub>. We conclude in Sect. 5 by summarizing our contributions and identifying avenues for future research.

## 2 Statement of Problems

In this section, we formally describe the problems under consideration and define the terms that will be used throughout the paper.

An linear program (LP) is a conjunction of linear inequalities and can be written in the form:  $\mathbf{A} \cdot \mathbf{x} \leq \mathbf{b}$ .

Throughout this paper, we use  $n$  to denote the number of variables in an LP and  $m$  to denote the number of constraints.

We now define several types of constraints referred to throughout this paper.

**Definition 1.** A constraint of the form  $a_i \cdot x_i \leq b_k$  is called an absolute constraint if  $a_i \in \{1, -1\}$  and  $b_k \in \mathbb{Z}$ .

**Definition 2.** A constraint of the form  $a_i \cdot x_i + a_j \cdot x_j \leq b_k$  is called a difference constraint, if  $a_i, a_j \in \{1, -1\}$ ,  $a_i = -a_j$ , and  $b_k \in \mathbb{Z}$ .

A conjunction of difference constraints is called a difference constraint system (DCS).

**Definition 3.** A constraint of the form  $a_i \cdot x_i + a_j \cdot x_j \leq b_k$  is called a Unit Two Variable per Inequality (UTVPI) constraint, if  $a_i, a_j \in \{0, 1, -1\}$ ,  $a_i$  and  $a_j$  are not both 0, and  $b_k \in \mathbb{Z}$ .

A conjunction of UTVPI constraints is called a UTVPI constraint system (UCS).

In the above definitions,  $b_k$  is called the defining constant of the constraint. Note that in this paper, we require  $b_k$  to be integral. Additionally, the terms  $x_i$  and  $-x_i$  are called literals.

*Example 1.*  $x_1 \leq 5$  is an absolute constraint,  $x_1 - x_2 \leq 4$  is a difference constraint, and  $x_1 + x_2 \leq 4$  is a UTVPI constraint.

Note that both absolute constraints and difference constraints are UTVPI constraints.

In this paper, we examine refutations that establish linear infeasibility. In LPs, we are interested in refutations that use the following inference rule:

$$\frac{\sum_{i=1}^n a_i \cdot x_i \leq b_1 \qquad \sum_{i=1}^n a'_i \cdot x_i \leq b_2}{\sum_{i=1}^n (a_i + a'_i) \cdot x_i \leq b_1 + b_2} \tag{1}$$

Rule (1) is called the addition (ADD) rule and corresponds to the summation of constraints. The ADD rule plays a role in the refutations of LPs that is similar to the role played by resolution in the refutations of CNF formulas. Observe that any assignment that satisfies the hypotheses of Rule (1) must also satisfy the consequent. Thus, Rule (1) is a **sound** inference rule.

Additionally, the completeness of the ADD rule was established by Farkas [6], in a lemma that is famously known as Farkas' Lemma for systems of linear inequalities [17]. Thus, if the input LP is unsatisfiable, then repeated applications of Rule (1) to the constraints of the LP will result in a contradiction of the form:  $0 \leq -b$ , where  $b > 0$ .

For systems of UTVPI constraints, Rule (1) can be restricted as follows:

$$\frac{a_i \cdot x_i + a_j \cdot x_j \leq b_{k_1} \qquad -a_j \cdot x_j + a_l \cdot x_l \leq b_{k_2}}{a_i \cdot x_i + a_l \cdot x_l \leq b_{k_1} + b_{k_2}} \tag{2}$$

We refer to Rule (2) as the *transitive inference rule*. Although it is a restricted version of the ADD rule, it remains both sound and complete for the purposes of proving the linear infeasibility of UCSs [12].

### 2.1 The Constraint-Required Read-Once Refutation (CROR) Problem

We now define what it means for a refutation to be read-once.

**Definition 4.** *A refutation is said to be read-once, if each constraint is used at most once in the derivation of a contradiction.*

This restriction applies to both constraints in the original system as well as those derived from previous inferences. However, a derived constraint can be reused if it can be rederived using a different set of input constraints. Note that not every UCS has a read-once refutation.

*Example 2.* Consider the UCS defined by System (3).

$$\begin{aligned}
 l_1 : x_1 + x_2 &\leq -2 & l_2 : -x_1 + x_4 &\leq 1 \\
 l_3 : -x_1 - x_4 &\leq 1 & l_4 : -x_2 + x_3 &\leq 0 \\
 l_5 : -x_2 - x_3 &\leq 0 & &
 \end{aligned} \tag{3}$$

Observe that  $l_1$  is the only constraint in System (3) with a negative defining constant. Thus,  $l_1$  must be included in *any* refutation of System (3).

Any refutation of System (3) must derive a constraint of the form  $0 \leq b$  where  $b < 0$ . Thus, all variables in  $l_1$  must be eliminated by using other constraints. To eliminate  $x_1$  from  $l_1$ , we must include either  $l_2$  or  $l_3$  in the refutation. However, if only one of these constraints is included, then the variable  $x_4$  is not eliminated. Thus, both  $l_2$  and  $l_3$  must be in the refutation.

Similarly, to eliminate  $x_2$  from  $l_1$ , we must include both  $l_4$  and  $l_5$ . If both constraints are not used, then the variable  $x_3$  is not eliminated.

Thus, any refutation of System (3) must include all five constraints in the system. However, the sum of these five constraints is the constraint  $l_6 : -x_1 - x_2 \leq 0$ . This is obviously not a contradiction. The only way to derive a contradiction is to include the constraint  $l_1$  a second time. Thus, System (3) does not have a read-once refutation.

However, every infeasible UCS has a refutation in which each constraint is used at most twice [19].

We study a variant of ROR known as the CROR problem.

**Definition 5.** *The Constraint-required ROR (CROR) problem in UCSs: Given a UCS  $\mathbf{U}$  and a set of constraints  $S \subseteq \mathbf{U}$ , does  $\mathbf{U}$  have a read-once refutation that uses all of the constraints in  $S$ ?*

### 2.2 The Minimum Weight Perfect Matching (MWPM) Problem

We briefly discuss the Minimum Weight Perfect Matching (MWPM) problem on undirected graphs. Let  $\mathbf{G} = \langle \mathbf{V}, \mathbf{E}, \mathbf{c} \rangle$  be an undirected graph, with vertex set  $\mathbf{V}$ , edge set  $\mathbf{E}$  and edge cost function  $\mathbf{c}$ . A *matching* is any collection of vertex-disjoint edges. A *perfect matching* is a matching in which each vertex  $v \in \mathbf{V}$  is matched. Without loss of generality, we assume that  $|\mathbf{V}|$  is even, since  $\mathbf{G}$  cannot have a perfect matching, otherwise.

**Definition 6.** *The MWPM<sub>D</sub> problem: Given a weighted, undirected graph  $\mathbf{G}$ , and integer  $L$ , does  $\mathbf{G}$  have a perfect matching with weight at most  $L$ ?*

In this paper, we relate the parallel complexity of the CROR problem in UCSs to the parallel complexity of the MWPM<sub>D</sub> problem.

### 2.3 Complexity Classes

We now define the complexity classes used in this paper. First, we define the complexity class **NC** [15].

**Definition 7.** *A problem belongs to the class **NC**, if it can be solved in polylogarithmic parallel time using a polynomial number of processors.*

If we relax the requirements to allow for the use of a quasi-polynomial number of processors, we end up with the class **Quasi-NC** [21].

**Definition 8.** *A problem belongs to the class **Quasi-NC (QNC)**, if it can be solved in polylogarithmic parallel time using a quasi-polynomial number of processors.*

Note that quasi-polynomial refers to functions in  $2^{O(\log^c n)}$ , for some constant  $c > 0$ .

The classes **RNC** and **Quasi-RNC** respectively generalize the above classes [14].

**Definition 9.** *A problem belongs to the class **RNC**, if there exists an algorithm that runs in polylogarithmic parallel time using a polynomial number of processors such that:*

1. On **yes** instances of the problem, the algorithm returns **true** with probability at least  $\frac{1}{2}$ .
2. On **no** instances of the problem, the algorithm always returns **false**.

**Definition 10.** *A problem belongs to the class **Quasi-RNC (QRNC)**, if there exists an algorithm that runs in polylogarithmic parallel time using a quasi-polynomial number of processors such that:*

1. On **yes** instances of the problem, the algorithm returns **true** with probability at least  $\frac{1}{2}$ .
2. On **no** instances of the problem, the algorithm always returns **false**.

For randomized algorithms, the concept of a pseudo deterministic algorithm has been defined [9].

**Definition 11.** *A randomized algorithm is **pseudo-deterministic**, if on a given instance  $S$ , it returns the same solution with high probability.*

For parallel algorithms, we have different ways to measure efficiency. These are known as work-optimality and work-efficiency [10].

**Definition 12.** *A parallel algorithm is **work-optimal**, if the total work done by the algorithm is within a constant factor of the work done by the best known sequential algorithm for the same problem.*

**Definition 13.** *A parallel algorithm is **work-efficient**, if the total work done by the algorithm is within a logarithmic factor of the work done by the best known sequential algorithm for the same problem.*

### 3 Motivation and Related Work

UTVPI constraints occur in a number of problem domains including but not limited to program verification, abstract interpretation, real-time scheduling, and operations research [12]. For systems of UTVPI constraints, the problem of checking for read-once refutations seems to be more difficult than that of checking for linear or even integer feasibility. Previous results have established that the problems of checking for linear feasibility [19] and integer feasibility [20] of a UCS can both be solved in  $O(m \cdot n)$  time. However, checking for the existence of read-once refutations takes  $O((m+n)^2 \cdot \log(m+n))$  time [18]. It is very important to note that this paper considers a **different** problem in that the read-once refutation is required to include a particular set of constraints. *Thus, this problem is a generalization of the problem considered in [18].*

In [12], the problem of checking the linear feasibility of a UCS was reduced to the problem of checking the linear feasibility of a DCS. Note that this problem is equivalent to the problem of finding shortest paths in a directed graph [3] and thus belongs to the class **NC** [13].

The  $MWPM_D$  problem is one of the classical problems in combinatorial optimization [11]. Over the years, there has been a steady stream of papers documenting improvements in algorithms for this problem [4,5,8]. While the  $MWPM_D$  problem is in **P**, it is unknown if the  $MWPM_D$  problem is in **NC**. Several papers have studied both this problem and its unweighted variant from the perspective of parallel algorithms. In [7], it was shown that the problem of checking if an unweighted bipartite graph has a perfect matching is in **QNC**. Additionally, it was shown in [9] that this problem can be solved pseudo-deterministically by an **RNC** algorithm. However, this same problem was shown to be in **NC** for planar graphs [1]. These results were then extended to the minimum weight perfect matching problem [2,16]. [2] also shows that the problem of **finding** a minimum weight perfect matching is equivalent to the  $MWPM_D$  problem under **NC** reductions.

In this paper, we demonstrate the equivalence between the CROR problem in UCSs and the  $MWPM_D$  problem from the perspective of efficient parallel computation. General matching problems have been extensively studied from a perspective of parallel complexity. However, it remains unknown if the  $MWPM_D$  problem belongs to the complexity class **NC** [2].

### 4 The CROR Problem in UTVPI Constraints

In this section, we show that the CROR problem in UTVPI constraints is **NC**-equivalent to the  $MWPM_D$  problem.

First, we reduce the CROR problem to the  $MWPM_D$  problem. This is done using a modified version of the reduction used in [18]. For the sake of completeness, we now describe that reduction.

Given a UCS  $\mathbf{U} : \mathbf{A} \cdot \mathbf{x} \leq \mathbf{b}$ , we construct the undirected graph  $\mathbf{G} = \langle \mathbf{V}, \mathbf{E}, \mathbf{c} \rangle$  as follows:

1. For each variable  $x_i$  in  $\mathbf{U}$ , add the vertices  $x_i^+$ ,  $x_i'^+$ ,  $x_i^-$ , and  $x_i'^-$  to  $\mathbf{V}$ . Additionally, add the edges  $x_i^- \overset{0}{-} x_i^+$  and  $x_i'^- \overset{0}{-} x_i'^+$  to  $\mathbf{E}$ .
2. Add the vertices  $x_0^+$  and  $x_0^-$  to  $\mathbf{V}$ . Additionally, add the edge  $x_0^- \overset{0}{-} x_0^+$  to  $\mathbf{E}$ .
3. For each constraint  $l_k$  of  $\mathbf{U}$ , add the vertices  $l_k$  and  $l'_k$  to  $\mathbf{V}$  and the edge  $l_k \overset{0}{-} l'_k$  to  $\mathbf{E}$ . Additionally:
  - (a) If  $l_k$  is  $x_i + x_j \leq b_k$ , add the edges  $x_i^+ \overset{\frac{b_k}{2}}{-} l_k$ ,  $x_i'^+ \overset{\frac{b_k}{2}}{-} l_k$ ,  $x_j^+ \overset{\frac{b_k}{2}}{-} l'_k$ , and  $x_j'^+ \overset{\frac{b_k}{2}}{-} l'_k$  to  $\mathbf{E}$ .
  - (b) If  $l_k$  is  $x_i - x_j \leq b_k$ , add the edges  $x_i^+ \overset{\frac{b_k}{2}}{-} l_k$ ,  $x_i'^+ \overset{\frac{b_k}{2}}{-} l_k$ ,  $x_j^- \overset{\frac{b_k}{2}}{-} l'_k$ , and  $x_j'^- \overset{\frac{b_k}{2}}{-} l'_k$  to  $\mathbf{E}$ .
  - (c) If  $l_k$  is  $-x_i + x_j \leq b_k$ , add the edges  $x_i^- \overset{\frac{b_k}{2}}{-} l_k$ ,  $x_i'^- \overset{\frac{b_k}{2}}{-} l_k$ ,  $x_j^+ \overset{\frac{b_k}{2}}{-} l'_k$ , and  $x_j'^+ \overset{\frac{b_k}{2}}{-} l'_k$  to  $\mathbf{E}$ .
  - (d) If  $l_k$  is  $-x_i - x_j \leq b_k$ , add the edges  $x_i^- \overset{\frac{b_k}{2}}{-} l_k$ ,  $x_i'^- \overset{\frac{b_k}{2}}{-} l_k$ ,  $x_j^- \overset{\frac{b_k}{2}}{-} l'_k$ , and  $x_j'^- \overset{\frac{b_k}{2}}{-} l'_k$  to  $\mathbf{E}$ .
  - (e) If  $l_k$  is  $x_i \leq b_k$ , add the edges  $x_i^+ \overset{\frac{b_k}{2}}{-} l_k$ ,  $x_i'^+ \overset{\frac{b_k}{2}}{-} l_k$ ,  $x_0^+ \overset{\frac{b_k}{2}}{-} l'_k$ , and  $x_0^- \overset{\frac{b_k}{2}}{-} l'_k$  to  $\mathbf{E}$ .
  - (f) If  $l_k$  is  $-x_i \leq b_k$ , add the edges  $x_i^- \overset{\frac{b_k}{2}}{-} l_k$ ,  $x_i'^- \overset{\frac{b_k}{2}}{-} l_k$ ,  $x_0^+ \overset{\frac{b_k}{2}}{-} l'_k$ , and  $x_0^- \overset{\frac{b_k}{2}}{-} l'_k$  to  $\mathbf{E}$ .

Observe that if  $\mathbf{U}$  has  $m$  constraints over  $n$  variables, then  $\mathbf{G}$  has  $(4 \cdot n + 2 \cdot m + 2)$  vertices and  $(2 \cdot n + 5 \cdot m + 1)$  edges. In other words,  $\mathbf{G}$  has  $O(m + n)$  vertices and  $O(m + n)$  edges.

As discussed in [18],  $\mathbf{G}$  has a negative weight perfect matching, if and only if,  $\mathbf{U}$  has a read-once refutation.

We now modify the above reduction to  $\mathbf{NC}$ -reduce the CROR problem in UCSs to the MWPM<sub>D</sub> problem.

Let  $\mathbf{U}$  be a UCS and let  $\mathbf{G}$  be the corresponding undirected graph. If  $S$  is a set of constraints in  $\mathbf{U}$ , let  $\mathbf{G}'_S$  be the graph constructed by removing the edge  $l_r \overset{0}{-} l'_r$  from  $\mathbf{G}$  for each constraint  $l_r \in S$ .

It is important to note that despite the similarities to the reduction in [18], this reduction reduces a different problem to the MWPM<sub>D</sub> problem.

We now establish the correctness of our reduction.

**Theorem 1.** *Let  $\mathbf{U}$  be a UCS and let  $S$  be a set of constraints in  $\mathbf{U}$ .  $\mathbf{U}$  has a read-once refutation that uses all of the constraints in  $S$ , if and only if,  $\mathbf{G}'_S$  has a negative weight perfect matching.*

*Proof.* First, assume that  $\mathbf{U}$  has a read-once refutation  $R$  that uses all of the constraints in  $S$ . Since  $R$  is a read-once refutation of  $\mathbf{U}$ , the corresponding undirected graph  $\mathbf{G}$  has a negative weight perfect matching  $M$  [18].

From [18], for each constraint  $l_k$  in  $\mathbf{U}$ , the perfect matching  $M$  uses the edge  $l_k - l'_k$ , if and only if,  $R$  does not use the constraint  $l_k$ . Let  $l_r$  be a constraint in  $S$ . Since  $R$  uses the constraint  $l_r$ , the edge  $l_r - l'_r$  is not in  $M$ . Note that this is true for each constraint in  $S$ . Thus,  $M$  is a negative weight perfect matching of the graph  $\mathbf{G}'_S$ .

Now assume that  $\mathbf{G}'_S$  has a negative weight perfect matching  $M$ . Note that  $M$  is also a negative weight perfect matching of  $\mathbf{G}$ . Thus,  $\mathbf{U}$  has a read-once refutation  $R$  [18].

For each constraint  $l_k$  in  $\mathbf{U}$ , the perfect matching  $M$  uses the edge  $l_k - l'_k$ , if and only if,  $R$  does not use the constraint  $l_k$ . Let  $l_r$  be a constraint in  $S$ . Since  $M$  is a perfect matching of  $\mathbf{G}'_S$ ,  $M$  does not use the edge  $l_r - l'_r$ . Thus,  $R$  uses the constraint  $l_r$  as desired.  $\square$

We now show that the graph  $\mathbf{G}'_S$  can be constructed efficiently in parallel.

**Theorem 2.** *Given a UCS  $\mathbf{U}$  with  $m$  constraints over  $n$  variables and a set  $S$  of constraints in  $\mathbf{U}$ , the corresponding graph  $\mathbf{G}'_S$  can be constructed in constant time using  $O(m + n)$  processors.*

*Proof.* The construction of  $\mathbf{G}'_S$  can be performed in parallel as follows:

1. For each  $i = 1 \dots n$ , the  $i^{th}$  processor creates the vertices and edges corresponding to the variable  $x_i$ . These are the vertices and edges specified in step (1) of the construction of  $\mathbf{G}$ . All of these edges are also in  $\mathbf{G}'_S$ . Note that, in this step, each processor creates four vertices and two edges. Additionally, no two processors are required to access the same memory locations. Thus, this step can be performed in constant time in the CREW PRAM model.
2. For each  $j = 1 \dots m$ , the  $j^{th}$  processor creates the vertices and edges corresponding to the constraint  $l_j$ . These are the vertices and edges specified in step (3) of the construction of  $\mathbf{G}$ . If  $l_j \notin S$ , then all of these edges are also in  $\mathbf{G}'_S$ . If  $l_j \in S$ , then the edge  $l_j - l'_j$  is not in  $\mathbf{G}'_S$ . Note that, in this step, each processor creates two vertices and four or five edges. Additionally, no two processors are required to access the same memory locations. Thus, this step can be performed in constant time in the CREW PRAM model.

From this, it is easy to see that the reduction described above from the CROR problem in UCSs to the MWPM $_D$  problem is an **NC** reduction.  $\square$

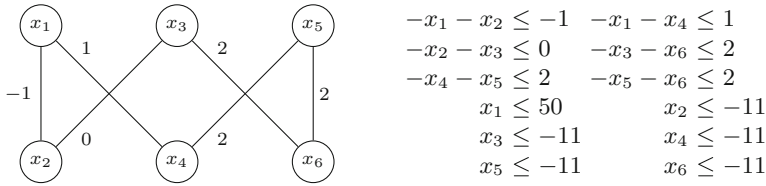
Note that this reduction works regardless of the size of  $S$ . Thus, the CROR problem can be **NC**-reduced to the MWPM $_D$  problem even when  $|S| \in O(m)$ .

Now we need to show that this reduction can be performed in the opposite direction. That is, we want to design an **NC** reduction from the MWPM $_D$  problem to the CROR problem in UCSs.



Let  $\mathbf{G}$  be an undirected graph with  $n$  vertices and  $m$  edges, and let  $L$  be an arbitrary integer. From  $\mathbf{G}$  and  $L$ , we construct a UCS  $\mathbf{U}$  as follows. 1. For each vertex  $x_i$  in  $\mathbf{G}$ , create the variable  $x_i$ . 2. For each edge  $x_i - x_j$  in  $\mathbf{G}$ , create the constraint  $-x_i - x_j \leq b_k$ . 3. Let  $-C$  be the smallest weight of any edge in  $\mathbf{G}$ . If all edge weights are positive, then let  $C = 0$ . Additionally, let  $W = \max\{m \cdot C + L + 1, 1\}$ . 4. Create the constraint  $x_1 \leq (n - 1) \cdot W - L - 1$ . 5. For each variable  $x_i, i = 2, \dots, n$ , create the constraint  $x_i \leq -W$ .

*Example 3.* Consider the undirected graph  $\mathbf{G}$  and corresponding UCS  $\mathbf{U}$ , when  $L = 4$ , in Fig. 1.



**Fig. 1.** Undirected Graph and corresponding UCS

Note that  $\mathbf{U}$  has a read-once refutation  $R$  that uses the constraint  $x_1 \leq 50$ .  $R$  consists of the constraints  $-x_1 - x_4 \leq 1, -x_2 - x_3 \leq 0, -x_5 - x_6 \leq 2, x_1 \leq 50$ , and  $x_2 \leq -11$  through  $x_6 \leq -11$ . Observe that  $\mathbf{G}$  has a perfect matching of weight at most 4. This matching consists of the edges  $x_1 - x_4, x_2 - x_3$ , and  $x_5 - x_6$ .

We now show that  $\mathbf{G}$  has a perfect matching of weight at most  $L$ , if and only if,  $\mathbf{U}$  has a read-once refutation that uses the constraint  $x_1 \leq (n - 1) \cdot W - L - 1$ .

**Theorem 3.** *Let  $\mathbf{G}$  be an undirected graph and let  $L$  be an arbitrary integer.  $\mathbf{G}$  has a perfect matching of weight at most  $L$ , if and only if, the corresponding UCS  $\mathbf{U}$  (constructed as per the discussion above) has a read-once refutation that uses the constraint  $x_1 \leq (n - 1) \cdot W - L - 1$ .*

*Proof.* Let  $M$  be a perfect matching in  $\mathbf{G}$  with cost  $c_M \leq L$ . From  $M$ , we can construct a read-once refutation  $R$  of  $\mathbf{U}$  as follows: 1. For each edge  $x_i - x_j$  in  $M$ , add the constraint  $-x_i - x_j \leq b_k$  to  $R$ . 2. Add  $x_1 \leq (n - 1) \cdot W - L - 1, x_2 \leq -W, \dots, x_n \leq -W$  to  $R$ . Note that summing the constraints  $x_1 \leq (n - 1) \cdot W - L - 1, x_2 \leq -W, \dots, x_n \leq -W$  results in the constraint  $\sum_{i=1}^n x_i \leq -L - 1$ .

Since  $M$  is a perfect matching, every vertex in  $\mathbf{G}$  is used by exactly one edge in  $M$ . Thus, each variable in  $\mathbf{U}$  is used by exactly one constraint of the form  $-x_i - x_j \leq b_k$  in  $R$ . Summing, these constraints results in the constraint  $-\sum_{i=1}^n x_i \leq c_M$ . Summing this result with the constraint  $\sum_{i=1}^n x_i \leq -L - 1$  results in the constraint  $0 \leq c_M - L - 1$ . Since  $c_M \leq L$ , this constraint is a

contradiction. Thus,  $R$  is a refutation of  $\mathbf{U}$ . Since  $R$  uses each constraint at most once,  $R$  is a read-once refutation as desired.

Now assume that  $\mathbf{U}$  has a read-once refutation  $R$  that uses the constraint  $x_1 \leq (n-1) \cdot W - L - 1$ . Any summation of the constraints in  $\mathbf{U}$  corresponding to the edges in  $\mathbf{G}$  results in a constraint with a defining constant  $H$  where  $H \geq -C \cdot m$  since there are  $m$  such constraints and the defining constant of each constraint is at least  $-C$ . If  $R$  uses  $f$  constraints of the form  $x_i \leq -W$ , then summing the constraints in  $R$  would result in a constraint of the form  $0 \leq (n-1-f) \cdot W - L - 1 + H$ , where  $H \geq -C \cdot m$ .

Since  $R$  is a read-once refutation,  $(n-1-f) \cdot W - L - 1 + H < 0$ . Note that  $W \geq C \cdot m + L + 1 \geq L + 1 - H$ . Thus,  $(n-1-f) \cdot W - L - 1 + H \geq (n-2-f) \cdot W$ . Since,  $(n-1-f) \cdot W - L - 1 + H < 0$  and  $W \geq 1$ ,  $(n-2-f) < 0$ . It follows that  $f \geq n-1$ . Consequently,  $R$  must use all constraints of the form  $x_i \leq -W$ . Summing these constraints, together with the constraint  $x_1 \leq (n-1) \cdot W - L - 1$  results in the constraint  $\sum_{i=1}^n x_i \leq -L - 1$ .

Since summing the constraints in  $R$  results in a contradiction of the form  $0 \leq b$  where  $b < 0$ , the remaining constraints in  $R$  must sum together to produce a constraint of the form  $-\sum_{i=1}^n x_i \leq c_M$  where  $c_M \leq L$ . By construction of  $\mathbf{U}$ , each  $-x_i$  term must come from a constraint of the form  $-x_i - x_j \leq b_k$ . Thus, the non-absolute constraints in  $R$  have the following properties: 1. Each variable  $x_i$  is used by exactly one constraint in  $R$  of the form  $-x_i - x_j \leq b_k$ . 2. The defining constants of these constraints sum to the value  $c_M \leq L$ .

Thus, the edges corresponding to these constraints form a perfect matching in  $\mathbf{G}$  with weight at most  $L$ .  $\square$

We now show that this is an **NC** reduction.

**Theorem 4.** *Given an undirected graph  $\mathbf{G}$  with  $n$  vertices and  $m$  edges the corresponding UCS  $\mathbf{U}$  can be constructed in  $O(\log n)$  time using  $O(m+n)$  processors.*

*Proof.* The construction can be performed in parallel as follows:

1. Find  $C$ . Note that this can be done in  $O(\log n)$  time using  $O(n)$  processors using a divide and conquer parallel search procedure.
2. For each  $j = 1 \dots m$ , the  $j^{\text{th}}$  processor creates the constraint corresponding to the  $j^{\text{th}}$  edge in  $\mathbf{G}$ . This is the constraint specified in step (2) of the construction of  $\mathbf{U}$ . Note that no two processors are required to access the same memory locations. Thus, this step can be performed in constant time in the CREW PRAM model.
3. For each  $i = 2 \dots n$ , the  $i^{\text{th}}$  processor creates the constraint  $x_i \leq -W$ . Meanwhile, the first processor creates the constraint  $x_1 \leq (n-1) \cdot W - L - 1$ . Note that, in this step, no two processors are required to access the same memory locations. Thus, this step can be performed in constant time in the CREW PRAM model.

From this, it is easy to see that the reduction from the MWPM<sub>D</sub> problem to the CROR problem in UCSs can be accomplished by an **NC** reduction.  $\square$

Note that there is only one constraint that is required to be used by the read-once refutation of  $\mathbf{U}$ . Thus, the  $\text{MWPM}_D$  problem can be  $\mathbf{NC}$ -reduced to the CROR problem in UCSs even when  $|S| = 1$ .

## 5 Conclusion

In this paper, we investigated the applicability of parallelization to the problem of finding CRORs in systems of UTVPI constraints. As mentioned previously, UTVPI constraints are an important class of linear constraints that find applications in a number of domains and hence short certificates of infeasibility (read-once refutations) are particularly useful.

In previous work [18], the ROR problem was reduced to the  $\text{MWPM}_D$  problem. We extended these results to a more restrictive form of refutation. Additionally, we were able to  $\mathbf{NC}$ -reduce the  $\text{MWPM}_D$  problem to the CROR problem in UTVPI constraints. Together, these reductions prove that the CROR problem in UCSs is  $\mathbf{NC}$ -equivalent to the  $\text{MWPM}_D$  problem. All of these reductions are designed using the CREW PRAM model of parallel computation.

From the perspective of future research, the following avenues appear promising:

1. In this paper, we related the parallel complexity of the CROR problem to the parallel complexity of the  $\text{MWPM}_D$  problem. However, we did not provide parallel algorithms for the ROR problem. Future research can focus on finding a parallel algorithm for the ROR problem in UCSs.
2. It is known that the minimum weight perfect matching problem for planar graphs belongs to the class  $\mathbf{NC}$ . Can this result be extended to provide an  $\mathbf{NC}$  algorithm for a restricted form of UTVPI constraint systems?
3. The problem of checking the linear feasibility of a UCS can be solved by an  $\mathbf{NC}$  algorithm for shortest paths. Can a similar result be obtained for checking the integer feasibility of a UCS? It is important to note that for integer feasibility, we need to use an additional inference rule.

**Acknowledgments.** This research was supported in part by the Air-Force Office of Scientific Research through grant FA9550-19-1-0177 and in part by the Air-Force Research Laboratory, Rome through Contract FA8750-17-S-7007.

This research was made possible by the NASA Established Program to Stimulate Competitive Research, Grant # 80NSSC22M0027.

## References

1. Anari, N., Vazirani, V.V.: Planar graph perfect matching is in NC. In: 2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS), pp. 650–661 (2017)
2. Anari, N., Vazirani, V.V.: Matching is as easy as the decision problem in the NC model. In 11th Innovations in Theoretical Computer Science Conference, ITCS. LIPIcs, vol. 151, pp. 54:1–54:25 (2020)

3. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: *Introduction to Algorithms*. MIT Press (2001)
4. Duan, R., Pettie, S., Su, H.-H.: Scaling algorithms for weighted matching in general graphs. *ACM Trans. Algorithms* **14**(1), 8:1–8:35 (2018)
5. Edmonds, J.: *An introduction to matching*, 1967. Mimeographed notes. Engineering Summer Conference, University of Michigan, Ann Arbor, MI
6. Farkas, G.: Über die Theorie der Einfachen Ungleichungen. *Journal für die Reine und Angewandte Mathematik* **124**(124), 1–27 (1902)
7. Fenner, S., Gurjar, R., Thierauf, T.: Bipartite perfect matching is in quasi-NC. In: *Proceedings of the Forty-eighth Annual ACM Symposium on Theory of Computing, STOC 2016*, pp. 754–763. ACM (2016)
8. Gabow, H.N.: An efficient implementation of Edmonds’ algorithm for maximum matching on graphs. *J. ACM* **23**(2), 221–234 (1976)
9. Goldwasser, S., Grossman, O.: Bipartite perfect matching in pseudo-deterministic NC. In: *44th International Colloquium on Automata, Languages, and Programming (ICALP)*. LIPIcs, vol. 80, pp. 87:1–87:13 (2017)
10. Khan, M., et al.: *Algorithms of Informatics*. Mondat Kft. Budapest, September 2013
11. Korte, B., Vygen, J.: *Combinatorial Optimization*. Number 21 in *Algorithms and Combinatorics*, 4th edn. Springer, New York (2010)
12. S. K. Lahiri and M. Musuvathi. An Efficient Decision Procedure for UTVPI Constraints. In *Proceedings of the 5<sup>th</sup> International Workshop on the Frontiers of Combining Systems*, September 19–21, Vienna, Austria, pages 168–183, New York, 2005. Springer
13. Leighton, F.T.: *Introduction to Parallel Algorithms and Architectures*. Morgan Kaufmann (1992)
14. Mulmuley, K., Vazirani, U.V., Vazirani, V.V.: Matching is as easy as matrix inversion. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing, STOC 1987*, pp. 345–354. Association for Computing Machinery, New York (1987)
15. Christos, H.: *Papadimitriou. Computational Complexity*. Addison-Wesley, New York (1994)
16. Sankowski, P.: NC Algorithms for Weighted Planar Perfect Matching and Related Problems. In: *45th International Colloquium on Automata, Languages, and Programming (ICALP)*. LIPIcs, vol. 107, pp. 97:1–97:16 (2018)
17. Schrijver, A.: *Theory of Linear and Integer Programming*. John Wiley and Sons, New York (1987)
18. Subramani, K., Wojciechowski, P.: A polynomial time algorithm for read-once certification of linear infeasibility in UTVPI constraints. *Algorithmica* **81**(7), 2765–2794 (2019)
19. Subramani, K., Wojciechowski, P.J.: A combinatorial certifying algorithm for linear feasibility in UTVPI constraints. *Algorithmica* **78**(1), 166–208 (2017)
20. Subramani, K., Wojciechowski, P.J.: A certifying algorithm for lattice point feasibility in a system of UTVPI constraints. *J. Comb. Optim.* **35**(2), 389–408 (2018)
21. Svensson, O., Tarnawski, J.: The matching problem in general graphs is in quasi-NC. In: *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 696–707 (2017)



# Extracting Densest Sub-hypergraph with Convex Edge-Weight Functions

Yi Zhou<sup>1,2</sup> , Shan Hu<sup>1</sup>, and Zimo Sheng<sup>1</sup>

<sup>1</sup> School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China  
zhou.yi@uestc.edu.cn, hu.shan@std.uestc.edu.cn

<sup>2</sup> Zhejiang Jinggong Steel Building Group, Shaoxing 312030, Zhejiang, China

**Abstract.** The densest subgraph problem (DSG) aiming at finding an induced subgraph such that the average edge-weights of the subgraph is maximized, is a well-studied problem. However, when the input graph is a hypergraph, the existing notion of DSG fails to capture the fact that a hyperedge partially belonging to an induced sub-hypergraph is also a part of the sub-hypergraph. To resolve the issue, we suggest a function  $f_e : \mathbb{Z}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$  to represent the partial edge-weight of a hyperedge  $e$  in the input hypergraph  $\mathcal{H} = (V, \mathcal{E}, f)$  and formulate a *generalized densest sub-hypergraph problem* (GDSH) as  $\max_{S \subseteq V} \frac{\sum_{e \in \mathcal{E}} f_e(e \cap S)}{|S|}$ . We demonstrate that, when all the edge-weight functions are non-decreasing convex, GDSH can be solved in polynomial-time by the linear program-based algorithm, the network flow-based algorithm and the greedy  $\frac{1}{r}$ -approximation algorithm where  $r$  is the rank of the input hypergraph. Finally, we investigate the computational tractability of GDSH where some edge-weight functions are non-convex.

**Keywords:** Densest subgraph problem · Hypergraph · Convex function

## 1 Introduction

The *densest subgraph problem* (DSG) is a well-known problem in research communities of operations research, combinatorial optimization, data mining and so on. Given an edge-weighted graph  $\mathcal{G} = (V, \mathcal{E}, w)$  with a vertex set  $V$ , an edge set  $\mathcal{E}$  and an edge-weight function  $w : \mathcal{E} \rightarrow \mathbb{R}_{\geq 0}$ , DSG asks us to maximize the density of the subgraph induced by a vertex set  $S \subseteq V$ , i.e.,  $\max_{S \subseteq V} \frac{\sum_{e \in \mathcal{E}} w(e)}{|S|}$ . Applications of DSG range from web community detection [6, 8], network motif clustering [3, 18] to information recommendation [17]. For solving DSG, there exists a network flow-based exact algorithm by Goldberg [5, 9], a linear program-based algorithm by Charikar [4] and a linear-time  $\frac{1}{2}$ -approximation algorithm in [1, 13].

This work is partially supported by the National Natural Science Foundation of China under Grant No. 61802049.

On the other hand, *hypergraph* is attracting increasing attentions in recent years. The hypergraph is a generalization of the normal graph in which a hyper-edge consists of arbitrary positive number of vertices. An edge-weighted hypergraph is defined as  $\mathcal{H} = (V, \mathcal{E}, w)$  where  $V$  is a vertex set,  $\mathcal{E}$  is a hyperedge set, and  $w : \mathcal{E} \rightarrow \mathbb{R}_{\geq 0}$  is an edge-weight function assigning each hyperedge a positive weight. The densest subgraph problem in an edge-weighted hypergraph, i.e. the *densest sub-hypergraph problem* (DSH), is known to be formulated as follows.

*Problem 1 (DSH as in [11, 18]).* Given a hypergraph  $\mathcal{H} = (V, \mathcal{E}, w)$ , DSH asks for a sub-hypergraph induced by  $S \subseteq V$  such that the average edge-weight of the sub-hypergraph i.e.,  $\frac{\sum_{e \subseteq S} w(e)}{|S|}$ , is maximized.

As far as we know, DSH was initially appeared as a generalization of the *densest  $r$ -clique problem (DrC)* in [18]. Given a graph  $\mathcal{G} = (V, \mathcal{E})$ , DrC asks for a subset of vertices  $S$  such that the average number of  $r$ -cliques (a clique of size  $r$ ) induced by  $S$  is maximized. Clearly, DrC can be reduced to DSH by building a  $r$ -uniform hypergraph in which each hyperedge represents a  $r$ -clique. For DrC, a polynomial exact algorithm and a  $\frac{1}{r}$ -approximation algorithm were introduced in [18], and a sampling algorithm was given in [16]. In [11], Hu *et al.* finally remove the assumption that the input hypergraph is  $r$ -uniform and formalize DSH as Problem 1. They demonstrated that the linear program, network flow and approximation algorithms for DSH. Recently, a much faster  $(1 - \epsilon)$  approximation algorithm based on max flow is given in [5] for DSH.

In this paper, we study a more generalized version of the densest sub-hypergraph problem rather than continue working with the existing model. We observed that in Problem 1, a hyperedge  $e$  is counted as a part of sub-hypergraph induced by  $S$  only when  $e$  is a subset of  $S$ . However, in some graph applications like [15], if a hyperedge  $e$  intersects with  $S$ , i.e.,  $e \not\subseteq S$  and  $e \cap S \neq \emptyset$ ,  $e$  is *partially* belong to the sub-hypergraph induced by  $S$ . Therefore, the fact that the weight of a sub-hypergraph induced by  $S$  should contain a partial weight of the hyper-edge that intersects with  $S$  is not captured by the definition of DSH. In order to fix this issue, we introduce an edge-weight function  $f_e : \{0, \dots, |e|\} \rightarrow \mathbb{R}_{\geq 0}$  for each hyperedge  $e$  in the input hypergraph, and then define the following *generalized densest sub-hypergraph problem (GDSH)*.

*Problem 2 (GDSH in this paper).* Given an edge-weighted hypergraph  $\mathcal{H} = (V, \mathcal{E}, f)$  with vertex set  $V$  and hyperedge set  $\mathcal{E}$ ,  $f_e : \{0, \dots, |e|\} \rightarrow \mathbb{R}_{\geq 0}$  being an edge-weight function for each  $e \in \mathcal{E}$ , GDSH asks for a set of vertices  $S \subseteq V$  such that i.e.,  $\frac{\sum_{e \in \mathcal{E}} f_e(|e \cap S|)}{|S|}$ , is maximized.

It is clear that GDSH generalizes the DSH problem. For example, if  $\mathcal{H} = (V, \mathcal{E}, w)$  is the input hypergraph for DSH, we can build a hypergraph  $\mathcal{H}' = (V, \mathcal{E}, f')$  such that  $f'_e(i) = 0$  when  $i < |e|$  and  $f'_e(|e|) = w(e)$ . Then, it is clear that the solution of GDSH with input graph  $\mathcal{H}'$  is the same as DSH with input graph  $\mathcal{H}$ . In this sense, GDSH also generalizes existing densest subgraph problems like DSG and DrCP.

Since convex functions are ubiquitous in many applications, in the remaining of the paper, we investigate GDSH with focus on cases where all edge-weight functions are *non-decreasing convex*. It is clear that all the above problems like DSG, DrC and DSH are special cases of GDSH with non-decreasing convex edge-weight functions. We will use  $n$  to denote the vertex number  $|V|$ ,  $m$  to denote the edge number  $|\mathcal{E}|$ ,  $p$  to denote  $\sum_{e \in \mathcal{E}} |e|$  and  $r$  to denote the rank  $\max_{e \in \mathcal{E}} |e|$  of the input hypergraph  $\mathcal{H} = (V, \mathcal{E}, f)$ . Note that  $p = \sum_{v \in V} \deg_{\mathcal{H}}(v)$  where  $\deg_{\mathcal{H}}(v)$  is the *degree* of vertex  $v \in V$ . We also use  $\Psi = \sum_{e \in \mathcal{E}} f_e(|e|)$  to denote the whole edge-weight of the hypergraph. Our main contributions for GDSH when all edge-weight functions are non-decreasing convex functions are summarized as follows.

- **A linear program whose optimal value is equal to the maximum density of GDSH.** The linear program has  $O(mr!)$  inequalities but efficient oracles exist for separation. We show that an optimal solution of GDSH can be easily obtained by solving the linear program.
- **A network flow-based algorithm which runs in  $O(\text{mincut}(p, pr) \log \Psi)$  time,  $\text{mincut}(N, M)$  representing the time of solving minimum  $s, t$ -cut in directed flow network with  $N$  vertices and  $M$  arcs.** We also show the technique to obtain a  $O(\text{mincut}(p, pr) \log(\epsilon^{-1} \log(rm)))$  time  $(1 - \epsilon)$  approximation algorithm which removes the  $\log \Psi$  factor.
- **A greedy  $\frac{1}{r}$  approximation algorithm with much faster running time  $O(pr \log n)$ .** With a little relaxation of the greedy strategy, the greedy approximation algorithm can also run in logarithmic iterations under the parallel computing settings.

It is worth mentioning that the above three algorithms extend the linear program algorithm, network flow algorithm and greedy algorithm, respectively, in [4, 11]. However, the extension is not trivial as. We only assume the non-decreasing and convexity properties of the edge-weight function in this work, contrary to the existing work that edge-weight functions are uniform and specifically given.

For completeness, we lastly study the computational tractability of GDSH when some edge-weight functions are non-convex. It turns out that when all edge-weight functions are non-decreasing concave, GDSH can be simply solved by selecting a (densest) vertex, when some edge-weight function are concave, GDSH is shown to be NP-hard by reduction from the max-cut problem.

## 2 Properties of Edge-Weight Functions

Given  $\mathcal{H} = (V, \mathcal{E}, f)$ , the edge weight function  $f_e$  is defined on discrete domain  $0, \dots, |e|$ . We first assume that  $f_e$  has *non-decreasing* properties for any  $e \in \mathcal{E}$ .

*Property 1 (Non-decreasing).*  $f_e(i) \leq f_e(i + 1), \forall i \in \{0, \dots, |e| - 1\}$

This property is a clearly natural in practice. Without loss of generality, we assume that  $f_e(0) = 0$ . If  $f_e(0) \neq 0$ , we can use  $f'_e(i) = f_e(i) - f_e(0)$  to replace  $f_e$  without changing the optimal solution of GDSH.

Aside from the non-decreasing property, we also discuss the *convexity* and *concavity* properties. As we know, convexity and concavity are common properties for many functions. They play important roles in characterizing the hardness of underlying optimization problems.

*Property 2 (Convexity).*

$$f_e(i) - f_e(i - 1) \leq f_e(i + 1) - f_e(i), \forall i \in \{1, \dots, |e| - 1\}.$$

*Property 3 (Concavity).*

$$f_e(i) - f_e(i - 1) \geq f_e(i + 1) - f_e(i), \forall i \in \{1, \dots, |e| - 1\}.$$

Given an  $S \subseteq V$ , we use  $F(S) = \sum_{e \in \mathcal{E}} f_e(|e \cap S|)$  to represent the *weight of sub-hypergraph induced by S*. Clearly, if  $\forall e \in \mathcal{E}$ ,  $f_e$  is non-decreasing convex (concave), then  $F(S)$  is a monotone *supermodular* (*submodular*) function in finite set  $V$  (because submodularity and supermodularity are closed under non-negative linear combination). Let us recall the definitions of *supermodularity* and *submodularity* as bellow.

*Property 4 (Supmodularity).*

$$F(S \cup \{v\}) - F(S) \leq F(T \cup \{v\}) - F(T), \forall S \subseteq T \subset 2^V \text{ and } \forall v \in S, v \notin T$$

*Property 5 (Submodularity).*

$$F(S \cup \{v\}) - F(S) \geq F(T \cup \{v\}) - F(T), \forall S \subseteq T \subset 2^V \text{ and } \forall v \in S, v \notin T$$

Lastly, we assume that  $f_e(i)$  is computed in constant time for any  $i \in \{0, \dots, |e|\}$ . Thus, for any set  $S \subseteq V$ ,  $f_e(|e \cap S|)$  is computed in time  $O(\min\{|e|, |S|\})$  and  $F(S)$  is computed in time  $O(p)$ .

### 3 GDSH with Convex Edge-Weight Functions

In this section, we investigate algorithms for solving GDSH when every edge-weight function is non-decreasing convex. Specifically, we show a linear program, a parametric network flow-based algorithm, and a fast greedy approximation in Sects. 3.1, 3.2 and 3.3, respectively.

#### 3.1 A Linear Program Approach

For a hyperedge  $e \in \mathcal{E}$ , let  $\mathcal{P}_e$  be the set of all permutations of  $e$ . Given a permutation  $\pi \in \mathcal{P}_e$ ,  $\pi(i) = v$  means that the  $i$ th vertex of permutation  $\pi$  is  $v$  and  $v \in e$ . Then, the linear program for GDSH, i.e., LP-GDSH, is given as follows.



$$\text{maximize } \sum_{e \in \mathcal{E}} y_e \quad (\text{LP-GDSH})$$

$$\text{s.t. } \sum_{i=1}^{|e|} (f_e(i) - f_e(i-1))x_{\pi(i)} \geq y_e \quad \forall e \in \mathcal{E}, \forall \pi \in \mathcal{P}_e \quad (1)$$

$$\sum_{v \in V} x_v \leq 1 \quad (2)$$

$$x_v \geq 0, y_e \geq 0 \quad \forall v \in V, e \in \mathcal{E}$$

**Lemma 1.** *Let  $\mathbf{x}$  be a feasible solution of LP-GDSH. Then, for any  $e \in \mathcal{E}$ , we have  $\min_{\pi \in \mathcal{P}_e} (\sum_{i=1}^{|e|} (f_e(i) - f_e(i-1))x_{\pi(i)}) = \sum_{i=1}^{|e|} (f_e(i) - f_e(i-1))x_{\pi^*(i)}$  where  $\pi^* \in \mathcal{P}_e$  is a permutation that  $x_{\pi^*(1)} \geq x_{\pi^*(2)} \geq \dots \geq x_{\pi^*(|e|)}$ .*

For compactness, we leave the proof of this lemma, as well as all the other missing proofs in the paper, to the appendix.

**Theorem 1.** *The following statements hold for LP-GDSH.*

1. *For any  $S \subseteq V$ , there is a feasible solution  $(\mathbf{x}, \mathbf{y})$  of LP-GDSH such that  $\sum_{e \in \mathcal{E}} y_e = \frac{F(S)}{|S|}$ .*
2. *Let  $\gamma^*$  be the optimal objective value of LP-GDSH. Then, there is a vertex set  $S \subseteq V$  such that  $\gamma^* \leq \frac{F(S)}{|S|}$ .*

*Therefore, the optimal solution of LP-GDSH is equal to the maximum density of GDSH.*

*Proof (sketch).* For the first statement, we can build a solution  $(\mathbf{x}, \mathbf{y})$  such that  $x_v = \frac{1}{|S|}$  if  $v \in S$  and  $x_v = 0$  otherwise, and  $y_e = \frac{f_e(|e \cap S|)}{|S|}$  for any  $e \in \mathcal{E}$ . Then, we show that this solution satisfies the statement. For the second statement, let us denote  $(\mathbf{x}^*, \mathbf{y}^*)$  as an optimal solution of LP-GDSH and  $S_r = \{v : x_v^* \geq r\}$ . We can show that there exists a  $r \in [0, 1]$  such that  $\gamma^* \leq \frac{F(S_r)}{|S_r|}$ . This last conclusion that the optimal value of LP-GDSH is equal to the maximum value of GDSH can be finally obtained by combining the two statements.

By the proof of Theorem 1, we can obtain the optimal solution to GDSH from the optimal LP solution  $(\mathbf{x}^*, \mathbf{y}^*)$  by simply solving  $\max_{r \in [0, 1]} \frac{F(S_r)}{|S_r|}$ . The number of inequalities in LP-GDSH is  $O(mr!)$ , but this linear program can be still solved in polynomial time because Inequality 1 can be efficiently separated by Lemma 1.

**Remark.** It is clear that LP-GDSH generalizes Charikar's linear program [4] for DSG and Hu's linear program [11] for DSH (Problem 1). A very recent work in [5] showed that the linear program technique can be also used for solving the densest supermodular subset problem which maximizes a supermodular set function of  $S$  over  $|S|$ . Our GDSH can be a special case of this problem as  $F(S)$  is supermodular if  $f_e$  is convex. It is also interested to see that our LP-GDSH can be reduced to their linear program by summarizing Inequality 1 over all  $e \in \mathcal{E}$ .

### 3.2 A Network Flow Algorithm

In this section, we introduce a parametric network flow-based approach algorithm, *GDSH-Flow*, for solving GDSH when  $f_e$  is non-decreasing convex. *GDSH-Flow* is a standard binary search algorithm which finds the optimal density within range  $[lb, ub]$ . Initially,  $lb = 0$  and  $ub = \Psi$ . *GDSH-Flow* testifies if there is a sub-hypergraph of density  $\lambda = \frac{lb+ub}{2}$  by computing  $\min_{S \subseteq V} (\gamma|S| - F(S))$ . If  $\min_{S \subseteq V} (\gamma|S| - F(S)) \leq 0$ , there exists a sub-hypergraph of density  $\lambda$ , then we set  $lb$  as  $\lambda$ . Otherwise, it indicates that  $\lambda$  is larger than the optimal, we then decrease  $ub$  to  $\lambda$ . In order to compute  $\min_{S \subseteq V} (\gamma|S| - F(S))$  for any  $\lambda$ , we make use of the minimum cut from a directed network flow  $\mathcal{G} = (U, \mathcal{A}, \lambda)$  where  $U$  and  $\mathcal{A}$  are vertex set and arc set, respectively.

---

**Algorithm 1:** Exact network flow algorithm for GDSH.

---

```

1 GDSH-Flow( $\mathcal{H}$ )
2 begin
3    $lb \leftarrow 0, ub \leftarrow \Psi$ 
4   while  $ub > lb$  do
5      $\lambda \leftarrow \frac{lb+ub}{2}$ 
6     Build directed flow network  $\mathcal{G} = (U, \mathcal{A}, \lambda)$ 
7     if the cost of min-cut  $(X, Y)$  in  $\mathcal{G}$  is larger than  $\Psi$  then
8        $ub \leftarrow \lambda$ 
9     else
10       $lb \leftarrow \lambda$ 
11   build directed flow network  $\mathcal{G} = (U, \mathcal{A}, lb)$ 
12   compute minimum cut  $(X, Y)$  from  $\mathcal{G}$ 
13   return  $X \cap V$ 

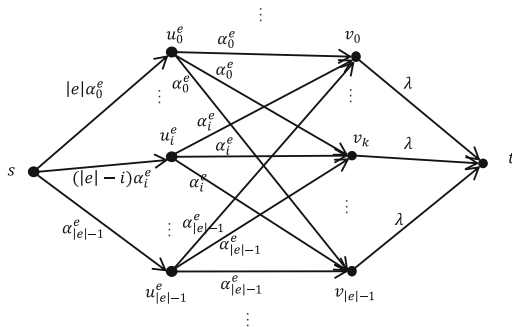
```

---

To illustrate how to build  $\mathcal{G} = (U, \mathcal{A}, \lambda)$ , we need to first assume that for any  $e \in \mathcal{E}$ ,  $f_e(i)$  returns integers for  $i = 0, \dots, |e|$ . This restriction does not impose any loss of generality because we can always obtain integer values by simultaneously scaling the edge-weight functions with an enough large value  $M$  which is a multiple of 10. Then,  $\mathcal{G}$  is built by the following steps.

- Build a source  $s$ , a sink  $t$  in  $\mathcal{G}$ , and make a copy of every vertex of  $V$  in  $\mathcal{G}$ .
- For a vertex  $v \in U \setminus \{s, t\}$ , add an arc  $(v, t)$  with capacity  $\lambda$  to  $\mathcal{G}$ .
- For a hyperedge  $e$  in  $\mathcal{H}$ , assume  $e = \{v_0, \dots, v_{|e|-1}\}$ . Then, add  $|e|$  vertices  $u_0^e, \dots, u_{|e|-1}^e$  to  $\mathcal{G}$ . Also, add the following arcs to  $\mathcal{G}$ .
  - For each  $i = 0, \dots, |e| - 1$ , add an arc  $(s, u_i^e)$  with capacity  $(|e| - i)\alpha_i^e$ .
  - For each  $i = 0, \dots, |e| - 1, j = 0, \dots, |e| - 1$ , add an arc  $(u_i^e, v_j)$  with capacity  $\alpha_i^e$  that

$$\alpha_i^e = \begin{cases} f_e(1) - f_e(0) & \text{if } i = 0, \\ f_e(i + 1) + f_e(i - 1) - 2f_e(i) & \text{if } 0 < i < |e|. \end{cases}$$



**Fig. 1.** An example of directed network  $\mathcal{G}$ .

An illustrative example of  $\mathcal{G}$  is shown in Fig. 1. Clearly,  $\alpha_i^e \geq 0$  for any integer  $0 \leq i < |e|$  because  $f_e$  is non-decreasing convex. We have the following statement for  $\mathcal{G}$ .

**Lemma 2.** *Let  $(X, Y)$  be a minimum  $s, t$ -cut in network  $\mathcal{G} = (U, A, \lambda)$  such that  $s \in X$  and  $t \in Y$ . Denote  $S = X \cap V$ . Then, the cost of  $(X, Y)$  is equal to  $\Psi + \lambda|S| - F(S)$ .*

*Proof.* First, by the definition of  $a_i^e$ , we have,

$$\begin{aligned} \sum_{i=0}^j \alpha_i^e &= f_e(1) - f_e(0) + \sum_{i=1}^{i=j} ((f_e(i+1) - f_e(i)) - (f_e(i) - f_e(i-1))) \\ &= f_e(j+1) - f_e(j) \end{aligned}$$

for any integer  $j < |e|$ .

Second, in network  $\mathcal{G}$ ,  $(X, Y)$  is a minimum  $s, t$ -cut,  $S = X \cap V$  and an edge  $e = \{v_0, \dots, v_{|e|-1}\}$ . Then, if  $|S \cap e| > i$ , then  $u_i^e \in X$ , otherwise,  $(X, Y)$  is not a minimum  $s, t$ -cut. In contrary, if  $|S \cap e| < i$ , then  $u_i^e \in Y$ . If  $|S \cap e| = i$ ,  $u_i^e$  can be either in  $X$  or  $Y$  without changing the cost of cut  $(X, Y)$ .

Let us denote  $s_e = |e \cap S|$  for simplicity. With the above observations, we finally get the cost of cut  $(X, Y)$  as

$$\begin{aligned} & \sum_{e \in \mathcal{E}} ((|e| - s_e) \sum_{i=0}^{s_e-1} a_i^e + \sum_{i=s_e}^{|e|-1} (|e| - i) a_i^e) + \lambda|S| \\ &= \sum_{e \in \mathcal{E}} (\sum_{i=0}^{s_e} a_i^e + \dots + \sum_{i=0}^{|e|-1} a_i^e) + \lambda|S| \\ &= \sum_{e \in \mathcal{E}} (f_e(s_e + 1) - f_e(s_e) + \dots + f_e(|e|) - f_e(|e| - 1)) + \lambda|S| \\ &= \sum_{e \in \mathcal{E}} (f_e(|e|) - f_e(s_e)) + \lambda|S| \\ &= \sum_{e \in \mathcal{E}} f_e(|e|) - F(S) + \lambda|S| \end{aligned}$$

, which ends the proof.

For a given  $\lambda \geq 0$ , Lemma 2 indicates that the cost of minimum  $s, t$ -cut  $(X, Y)$  is  $\Psi + \min_{S \subseteq V} (\lambda|S| - F(S))$ . Thus, we can decide whether there exists an  $S \subseteq V$  such that  $\lambda|S| - F(S) < 0$  by checking whether the cost of minimum  $s, t$ -cut of  $\mathcal{G}$  is smaller than  $\Psi$ . Therefore, the correctness of Alg. 1 is straightforward due to Lemma 2.

**Theorem 2.** *If  $\forall e \in \mathcal{E}$  in  $\mathcal{H} = (V, \mathcal{E}, f)$ ,  $f_e$  is a non-decreasing convex function, then Alg. 1 solves GDSH in  $O(\text{mincut}(p, pr) \log \Psi)$  time where  $\text{mincut}(N, M)$  is the time of finding minimum  $s, t$ -cut from a directed flow graph with  $N$  vertices and  $M$  edges.*

For any parameter  $\lambda$ , the number of vertices and edges in  $\mathcal{G} = (U, \mathcal{A}, \lambda)$  is  $n+p+2$  and  $n+p+pr$ , respectively. Therefore, the running time of this flow based algorithm is  $O(\text{mincut}(p, pr) \log \Psi)$ . For example, if we use the minimum  $s, t$ -cut algorithm in [10], which has running time  $O(NM \log \frac{N^2}{M})$  and space  $O(M)$ , the flow based algorithm runs in time  $O(npr \log(\frac{(n+p)^2}{n+p+pr}) \log \Psi)$  and space  $O(pr)$ .

**Remark.** Readers who are familiar with submodular optimization can realize that  $h_\lambda(S)$  is monotone submodular when  $f_e$  is non-decreasing convex for any  $e \in \mathcal{E}$ . Therefore  $\min_{S \subseteq V} h_\lambda(S)$  can be also solved via *Submodular Function Minimization* algorithms. The best-known submodular function minimization algorithm runs in time  $O(N^3(\log^2 N)EO + N^4 \text{polylog}(N))$  where  $N$  is the number of elements and  $EO$  is the maximum time of evaluating the submodular function [14]. In our case,  $N = n$  and  $EO = O(p)$ , the overall time is  $O((n^3p \log^2 n + n^4 \text{polylog}(n)) \log \Psi)$  which is not as efficient as our the network flow based approach.

**Further Removing the  $\log \Psi$  Factor.** Inspired by the technique in [12], we can obtain an algorithm with time polynomial to the size of input graph and  $\epsilon^{-1}$  by a little modification of *GDSH-Flow*. The algorithm, as shown in Algorithm 3 in the appendix, is named *GDSH-Flow- $\epsilon$* , which is  $(1 - \epsilon)$  approximation.

**Theorem 3.** *If  $\forall e \in \mathcal{E}$  in  $\mathcal{H} = (V, \mathcal{E}, f)$ ,  $f_e$  is a non-decreasing convex function, then *GDSH-Flow- $\epsilon$*  is a  $(1 - \epsilon)$  approximation algorithm with running time  $O(\text{mincut}(p, pr) \log(\epsilon^{-1} \log(rm)))$  for DHSP.*

### 3.3 A Fast $\frac{1}{r}$ -approximation Algorithm

We further introduce *GDSH-Approx* in Algorithm 2 to approximate GDSH when all the edge-weight functions are non-decreasing convex. By a little sacrifice on the accuracy, *GDSH-Approx* is much faster than the above approaches.

*GDSH-Approx* maintains a subset of vertices  $S$ . In each of the consequent iterations, *GDSH-Approx* identifies  $v$ , a vertex by which is removed from  $S$ , the decrease to the total edge-weight of the sub-hypergraph induced by  $S$  is minimized. The algorithm starts with  $S = V$  and stops when  $S$  becomes empty. Of all the sets  $S$  during the iterations, the one maximizing  $\frac{F(S)}{|S|}$  is returned. To shown the approximate ratio of *GDSH-Approx*, we first need the following observation.

**Algorithm 2:** Find the densest hyper-subgraph approximately.

```

1 GDSH-Approx( $\mathcal{H}$ )
2 begin
3    $S \leftarrow V, S' \leftarrow V$ 
4   for  $S \neq \emptyset$  do
5     Find  $v \in \operatorname{argmin}_{v \in S} (F(S) - F(S \setminus \{v\}))$ 
6      $S \leftarrow S \setminus \{v\}$ 
7     if  $\frac{F(S)}{|S|} > \frac{F(S')}{|S'|}$  then
8        $S' \leftarrow S$ 
9   return  $S'$ 

```

**Lemma 3.** Given any  $S \subseteq V$  in hypergraph  $\mathcal{H} = (V, \mathcal{E}, f)$ ,  $F(S) \geq \frac{1}{r} \sum_{u \in S} (F(S) - F(S \setminus \{u\}))$ .

Then, we have the following result for *GDSH-Approx*.

**Theorem 4.** If  $\forall e \in \mathcal{E}$  in  $\mathcal{H} = (V, \mathcal{E}, f)$ ,  $f_e$  is a non-decreasing convex function, *GDSH-Approx* is a  $\frac{1}{r}$ -approximate algorithm.

*Proof.* Assume  $S^* \subseteq V$  is a set of density  $\lambda^*$  in  $\mathcal{H}$ . Due to the optimality of  $S^*$ , for any  $v \in S^*$ ,

$$\lambda^* = \frac{F(S^*)}{|S^*|} \geq \frac{F(S^* \setminus \{v\})}{|S^*| - 1} = \frac{F(S^*) - (F(S^*) - F(S^* \setminus \{v\}))}{|S^*| - 1}.$$

with simple elementary transformations of the above inequality, we have  $F(S^*) - F(S^* \setminus \{v\}) \geq \lambda^*$ .

Now, let us consider the iteration of *GDSH-Approx* before the first vertex of  $S^*$ , say  $v$ , is removed. Call the current set of this iteration  $S'$ . So,  $S^* \subseteq S'$ . We have

$$\begin{aligned} \forall u \in S', F(S') - F(S' \setminus \{u\}) &\geq F(S') - F(S' \setminus \{v\}) \\ &\geq F(S^*) - F(S^* \setminus \{v\}) \geq \lambda^* \end{aligned}$$

where the first inequality follows from greedy strategy in the algorithm and the second inequality follows from the supermodularity of  $F(S)$  (since all edge-weight functions are convex). Now, combining Lemma 3, we conclude that

$$F(S') \geq \frac{1}{r} \sum_{u \in S'} (F(S') - F(S' \setminus \{u\})) \geq \sum_{u \in S'} \frac{\lambda^*}{r} = \frac{|S'| \lambda^*}{r}$$

Therefore,  $\frac{F(S')}{|S'|} \geq \frac{\lambda^*}{r}$ . Since the algorithm returns a set of maximum density of all the iterations, the approximation ratio follows.

The number of iterations of Algorithm 2 is  $n$ , the time to evaluate  $F(S)$  is  $O(p)$  in each iteration. Therefore, the running time of a simple implementation of this algorithm can be  $O(n^2p)$ . Using a minimum-heap to [7] to maintain the vertices in  $S$ , we can reduce the time to  $O(pr \log n)$ .

**Further Reducing the Number of Iterations.** Currently, the number of iterations of *GDSH-Approx* is clearly  $\Theta(n)$ . Motivated by the work in [2], we provide a method of revising Algorithm 2 such that the number of iterations reduces to the logarithmic scale. The new approximation algorithm is called *GDSH-Para* which is described in Algorithm 4 in the appendix. *GDSH-Para* would be very efficient in processing large hypergraphs in the parallel processing system because it only have a small number of dependable iterations.

**Theorem 5.** *If  $\forall e \in \mathcal{E}$  in  $\mathcal{H} = (V, \mathcal{E}, f)$ ,  $f_e$  is a non-decreasing convex function, then *GDSH-Para* is a  $\frac{1}{r(1+\epsilon)}$ -approximation with  $O(\log_{1+\epsilon} n)$  iterations.*

Like *GDSH-Approx*, the space consumption of *GDSH-Para* is  $O(n)$  if the minimum-heap data structure is used.

### 4 Non-convex Edge-Weight Functions

In this section, we investigate GDSH when some of the edge-weight functions are not non-decreasing convex.

**Theorem 6.** *If  $\forall e \in \mathcal{E}$  in  $\mathcal{H} = (V, \mathcal{E}, f)$ ,  $f_e$  is a non-decreasing concave function, the solution of *GDSH-Approx* is  $\{v\}$  where  $v = \operatorname{argmax}_{u \in V} \sum_{e \in \mathcal{E}: u \in e} f_e(1)$ .*

*Proof.* If for every  $e \in \mathcal{E}$ ,  $f_e$  is non-decreasing concave, then  $F(S)$  is a monotone submodular function. Besides,  $F(\emptyset) = 0$  because  $f_e(0) = 0$ . We first claim that, for any unit vertex set  $S$  that  $|S| = 1, S \subseteq T \subseteq V, F(S) \geq \frac{F(T)}{|T|}$  holds. To verify the claim, let us assume  $S = \{v_1\}$  and  $T = \{v_1, \dots, v_p\}$  without loss of generality, where  $p \geq 1$  is the size of  $T$ . By submodularity, we have

$$\begin{aligned} F(T) - F(\{v_1, \dots, v_{p-1}\}) &\leq F(S) - F(\emptyset) \\ F(\{v_1, \dots, v_{p-1}\}) - F(\{v_1, \dots, v_{p-2}\}) &\leq F(S) - F(\emptyset) \\ \dots & \\ F(\{v_1\}) - F(\emptyset) &\leq F(S) - F(\emptyset) \end{aligned}$$

By adding up the above inequalities, we obtain  $F(T) - F(\emptyset) \leq p(F(S) - F(\emptyset))$ . As  $F(\emptyset) = 0$ , we have  $F(S) \geq \frac{F(T)}{p} = \frac{F(T)}{|T|}$

Now, it is not hard to see that the optimal solution to GDSH is a set with one vertex  $v = \operatorname{argmax}_{u \in V} \sum_{e \in \mathcal{E}: u \in e} f_e(1)$ .

On the other hand, if there are some edge-weight function  $f_e$  that is (non-monotonic) concave in  $\mathcal{H} = (V, \mathcal{E}, f)$ , then we have the following NP-hardness result.

**Theorem 7.** *Given a hypergraph  $\mathcal{H} = (V, \mathcal{E}, f)$ , if for some  $e \in \mathcal{E}$   $f_e$  is concave and for other  $e \in \mathcal{E}$ ,  $f_e$  is non-decreasing convex, then *GDSH* is NP-hard.*

## 5 Conclusion

It is known that the (edge-weighted) densest sub-hypergraph problem is important in many data-mining applications. In this paper, we studied this problem with respect to different properties of the edge weight functions and formalized the Generalized Densest Sub-Hypergraph problem (GDSH). We show that GDSH with non-decreasing convex edge-weight functions can be solved efficiently by a linear program-based approach, a network flow-based approach and a fast greedy approximation algorithm. We also investigated GDSH for some other cases where edge-weight function are not always non-decreasing convex.

In the future, it would be interesting to extend the study from multiple dimensions. First, one could consider more properties about the edge weight functions like submodularity, or, one could also investigate faster algorithms when the edge-weight functions are identical. Besides, the GDSH problem under some constraints would be another interesting topic. For example, the problem of finding densest subgraph with at least  $k$  vertices is NP-hard, but 2-approximated was given in [13]. So, it could be possible to investigate the GDSH with different size constraint.

## A Missing Proof of Lemma 1

*Proof.* We justify the equation by contradiction. Assume that  $\pi' \in \mathcal{P}_e$  is a minimum permutation, i.e.,  $\sum_{i=1}^{|e|} (f_e(i) - f_e(i-1))x_{\pi'(i)} = \min_{\pi \in \mathcal{P}_e} \sum_{i=1}^{|e|} (f_e(i) - f_e(i-1))x_{\pi(i)}$ , but there exists  $i, j$  that  $1 \leq i < j \leq |e|$ ,  $x_{\pi'(i)} < x_{\pi'(j)}$ . As  $f_e$  is a non-decreasing convex function, we have  $0 \leq f_e(i) - f_e(i-1) \leq f_e(j) - f_e(j-1)$ . Then, we have  $(f_e(i) - f_e(i-1))x_{\pi'(j)} + (f_e(j) - f_e(j-1))x_{\pi'(i)} < (f_e(i) - f_e(i-1))x_{\pi'(i)} + (f_e(j) - f_e(j-1))x_{\pi'(j)}$ . In other words, we can decrease  $\sum_{i=1}^{|e|} (f_e(i) - f_e(i-1))x_{\pi'(i)}$  by exchanging  $\pi'(i)$  and  $\pi'(j)$ , which contradicts the assumption that  $\pi'$  is the minimum permutation.

## B Detailed Proof of Theorem 1

*Proof. Proof of the first statement.* For any  $S \subseteq V$ , we construct a  $\mathbf{x}$  such that  $x_v = \frac{1}{|S|}$  if  $v \in S$  and  $x_v = 0$  otherwise. Clearly,  $\mathbf{x}$  satisfies Inequality 2. We also construct  $\mathbf{y}$  with  $y_e = \frac{f_e(|e \cap S|)}{|S|}$  for any  $e \in \mathcal{E}$ . Then  $\sum_{e \in \mathcal{E}} y_e$  is equal to  $\frac{F(S)}{|S|}$ . Now, let use verify that this  $(\mathbf{x}, \mathbf{y})$  satisfies Inequality 1. By Lemma 1, the left-hand side of Inequality 1 is at least  $\sum_{i=1}^{|e|} (f_e(i) - f_e(i-1))x_{\pi^*(i)}$  where  $\pi^* \in P_e$  satisfies  $\pi^*(i) > \pi^*(i+1)$  for  $1 \leq i \leq n-1$ . Therefore,

$$\begin{aligned}
 & \sum_{i=1}^{|e|} (f_e(i) - f_e(i-1))x_{\pi(i)} \\
 & \geq \sum_{i=1}^{|e|} (f_e(i) - f_e(i-1))x_{\pi^*(i)} \\
 & = \sum_{i=1}^{|e \cap S|} (f_e(i) - f_e(i-1)) \frac{1}{|S|} \\
 & = \frac{f_e(|e \cap S|)}{|S|} \\
 & = y_e.
 \end{aligned}$$

Hence, the first statement holds.

**Proof of the Second Statement.** Let  $(\mathbf{x}^*, \mathbf{y}^*)$  be an optimal solution of LP-GDSH. Define  $S_r = \{v : x_v^* \geq r\}$ . We claim that there exists  $r \in [0, 1]$  such that  $\frac{F(S_r)}{|S_r|} \geq \gamma^*$ . Assume that there is no such  $r$ . Then we have  $F(S_r) < \gamma^* |S_r|$  for any  $r \in [0, 1]$ . That is to say,  $\int_0^\infty F(S_r)dr < \gamma^* \int_0^\infty |S_r|dr$ .

On the other hand, we have

$$\gamma^* \int_0^1 |S_r|dr = \gamma^* \sum_{v \in V} x_v^*$$

and

$$\begin{aligned}
 \int_0^1 F(S_r)dr &= \sum_{e \in \mathcal{E}} \int_0^1 f_e(|e \cap S|)dr \\
 &= \sum_{e \in \mathcal{E}} \left( \sum_{i=1}^{|e|} (f_e(i) - f_e(i-1))x_{\pi^*(i)} \right) \\
 &= \sum_{e \in \mathcal{E}} y_e^*.
 \end{aligned}$$

Note that the last equation is from the fact that  $y_e^*$  is equal to the minimum of  $\sum_{i=1}^{|e|} (f_e(i) - f_e(i-1))x_{\pi(i)}^*$  for any permutation  $\pi \in \mathcal{P}_e$ .

Hence, we have  $\sum_{e \in \mathcal{E}} y_e^* < \gamma^* \sum_{v \in V} x_v^*$  by assumption. However, this contradicts the condition that  $\gamma^*$  is optimal value. Therefore, we conclude that we can definitely find a  $r \in [0, 1]$  such that  $\frac{F(S_r)}{|S_r|} \geq \gamma^*$ .

### C Missing Proof of Lemma 3

*Proof.* For any hyperedge  $e$ , it is clear that if vertex  $u \in e \cap S$ ,  $f_e(|e \cap S|) - f_e(|e \cap (S \setminus \{u\})|) \leq f(|e \cap S|)$  and if  $u \notin e \cap S$ ,  $f_e(|e \cap S|) - f_e(|e \cap (S \setminus \{u\})|) = 0$ .



Hence, the following inequality holds.

$$\sum_{u \in S} (f_e(|e \cap S|) - f_e(|e \cap (S \setminus \{u\})|)) \leq |S \cap e| f_e(|e \cap S|) \leq r f_e(|e \cap S|)$$

By summarizing the above inequalities for all  $e \in \mathcal{E}$ , we have

$$\sum_{u \in S} (F(S) - F(S \setminus \{u\})) \leq r F(S)$$

which completes the proof.

## D Missing Proof to Theorem 3

*Proof.* The approximation ratio is clearly guaranteed by the stop condition of the algorithm. We mainly show that the number of *while* iterations is bounded by  $\log(\epsilon^{-1} \log(rm))$ . The crux is that  $\frac{ub}{lb}$  is shrunk by a square root after every iteration. Let  $i \in 1, \dots, i^*$  denote the iteration number of the algorithm,  $i^*$  is the last iteration number. In the  $i$ th iteration, let  $lb^i$  and  $ub^i$  be the lower and upper bound respectively.

First, it is clear that  $\frac{ub^1}{lb^1} = \frac{|e_m| \Psi}{f_{e_m}(|e_m|)} \leq rm$ . Then, we have

$$\begin{aligned} \frac{ub^{i+1}}{lb^{i+1}} &\leq \max\left(\frac{ub^i}{\lambda^i}, \frac{\lambda^i}{lb^i}\right) \\ &= \max\left(\frac{ub^i}{\sqrt{lb^i * ub^i}}, \frac{\sqrt{lb^i * ub^i}}{lb^i}\right) \\ &= \sqrt{\frac{ub^i}{lb^i}} \end{aligned}$$

Hence, we have  $\frac{ub^{i+1}}{lb^{i+1}} \leq \left(\frac{ub^1}{lb^1}\right)^{\frac{1}{2^i}}$ . On the other hand, we have  $\frac{ub^{i^*}}{lb^{i^*}} \leq \frac{1}{1-\epsilon}$ . Therefore,  $i^* \in O(\log(\frac{\log(rm)}{\log(\frac{1}{1-\epsilon})}))$ . As  $\lim_{\epsilon \rightarrow 0} \frac{\epsilon}{\log(\frac{1}{1-\epsilon})} = 1$ , we have  $i^* \in O(\log(\epsilon^{-1} \log(rm)))$ . Therefore, the overall running time of is bounded by  $O(\text{mincut}(p, np) \log(\epsilon^{-1} \log(rm)))$ .

## E Missing Proof of Theorem 5

*Proof.* By proof of Theorem 4, for any vertex  $v$  in a optimal solution  $S^*$ ,  $F(S^*) - F(S^* \setminus \{v\}) \geq \lambda^*$  where  $\lambda^*$  is the maximum density. Let us consider the pass before a first vertex  $v$  from  $S^*$  is removed in the algorithm. Denote the set as  $S'$ . Similar to the proof of Theorem 4, we still have  $F(S') - F(S' \setminus \{v\}) \geq F(S^*) - F(S^* \setminus \{v\}) \geq \lambda^*$  due to the supermodularity of  $F$  and the optimality of  $S^*$ . Then,

$$\frac{F(S')}{|S'|} \geq \frac{F(S') - F(S' \setminus \{v\})}{r(1 + \epsilon)} \geq \frac{\lambda^*}{r(1 + \epsilon)}$$

where the first inequality is a direct result of the strategy in Line 5 in the algorithm. Hence, we obtain the approximation ratio.

We now estimate the maximum number of iterations. At each iteration, for the current set  $S$ ,

$$\begin{aligned} F(S) &\geq \sum_{u \in S} \frac{F(S) - F(S \setminus \{u\})}{r} \\ &= \sum_{u \in \Delta} \frac{F(S) - F(S \setminus \{u\})}{r} + \sum_{u \in S \setminus \Delta} \frac{F(S) - F(S \setminus \{u\})}{r} \\ &> r(1 + \epsilon) \frac{F(S)}{|S|} \cdot \frac{|S \setminus \Delta|}{r} \end{aligned}$$

where the first inequality follows from Lemma 3, the second follows from the fact that any  $u \in S \setminus \Delta$  satisfies  $F(S) - F(S \setminus \{u\}) > r(1 + \epsilon) \frac{F(S)}{|S|}$ . Thus,  $|S \setminus \Delta| < \frac{1}{1+\epsilon}|S|$ , indicating that the size of  $S$  decreases by a factor at least  $\frac{1}{1+\epsilon}$  during each iteration. Therefore, the algorithm stops in  $O(\log_{1+\epsilon} n)$  iterations.

## F Missing Proof of Theorem 7

*Proof.* We reduce the well-known NP-hard problem, max-cut, to GDSH that edge-weight functions contain both convex and concave functions.

Given an unweighted graph  $\mathcal{H} = (V, \mathcal{E})$  where  $n = |V|$ , the max-cut problem asks to find  $T \subseteq V$  such that  $cut_{\mathcal{H}}(T) = |\{e \in \mathcal{E} : |e \cap T| = 1\}|$  is maximized. To show the reduction, we build an edge-weighted hypergraph  $\mathcal{H}^* = (V^*, \mathcal{E}^*, f^*)$  which includes both concave and convex edge-weight functions.

- Make two disjoint copies of  $\mathcal{H}$  of the same vertex and (hyper)edge sets. Denote the two copies as  $\mathcal{H}' = (V', \mathcal{E}', f')$  and  $\mathcal{H}'' = (V'', \mathcal{E}'', f'')$ . For each hyperedge  $e$  in both  $\mathcal{H}'$  and  $\mathcal{H}''$ , set  $f_e(i) = 1$  if  $i = 1$  and  $f_e(i) = 0$  for all  $i \neq 1$ .
- For vertex  $v \in V$ , insert a hyperedge  $e_v = \{v', v''\}$  where  $v'$  and  $v''$  are the two copies of  $v$  in  $\mathcal{H}'$  and  $\mathcal{H}''$ , respectively. For each hyperedge  $e_v$ , set  $f_{e_v}(i) = n^2$  if  $i = 1$  and  $f_{e_v}(i) = 0$  for all  $i \neq 1$ . Denote the set of these hyperedges as  $\mathcal{E}'''$ .
- Add a hyperedge  $e_n = V' \cup V''$  and assign the edge-weight function

$$f_{e_n}(i) = \begin{cases} 0, & \text{if } 0 \leq i < n \\ n^2(i - n + 1), & \text{if } n \leq i \leq 2n. \end{cases}$$

to hyperedge  $e_n$ .

In summary,  $\mathcal{H}^*$  includes a set of  $2n$  vertices and four sets of hyperedges,  $\mathcal{E}'$ ,  $\mathcal{E}''$ ,  $\mathcal{E}'''$  and  $\{e_n\}$ . The edge-weights functions of hyperedges in  $\mathcal{E}'$ ,  $\mathcal{E}''$ ,  $\mathcal{E}'''$  are concave but  $f_{e_n}$  is convex. Given  $\mathcal{H}^*$ , GDSH is to find a set  $S \subseteq V^*$  such that  $\frac{F(S)}{|S|} = \frac{cut_{\mathcal{H}'}(S) + cut_{\mathcal{H}''}(S) + n^2|\{e \in \mathcal{E}''' : |e \cap S| = 1\}| + f_{e_n}(|S|)}{|S|}$  is maximized. (To be precise,  $cut_{\mathcal{H}'}(S)$  and  $cut_{\mathcal{H}''}(S)$  are the abbreviations of  $cut_{H'}(S \cap V')$  and  $cut_{H''}(S \cap V'')$ , respectively.)

Suppose that  $S^*$  is an optimal solution of GDSH in  $H^*$ . We first demonstrate that the size of  $S^*$  is equal to  $n$ . Assume  $|S^*| < n$ .

$$\begin{aligned} \frac{F(S^*)}{|S^*|} &= \frac{cut_{\mathcal{H}'}(S^*) + cut_{\mathcal{H}''}(S^*) + n^2|\{e \in \mathcal{E}''' : |e \cap S^*| = 1\}| + 0}{|S^*|} \\ &\leq \frac{cut_{\mathcal{H}'}(S^*) + cut_{\mathcal{H}''}(S^*) + n^2|S^*|}{|S^*|} \\ &\leq \frac{cut_{\mathcal{H}'}(S^*) + cut_{\mathcal{H}''}(S^*)}{|S^*|} + n^2 \\ &< \frac{n \cdot |S^*|}{|S^*|} + n^2 \\ &= n + n^2 \end{aligned}$$

On the other hand, if  $|S^*| = n$ ,  $\frac{F(S^*)}{|S^*|} = n^2 + n + \frac{cut_{\mathcal{H}'}(S^*) + cut_{\mathcal{H}''}(S^*)}{n}$  which is not smaller than  $n + n^2$ . Hence,  $|S^*| \geq n$ .

Now suppose  $|S^*| > n$ . Then,

$$\begin{aligned} \frac{F(S^*)}{|S^*|} &= \frac{cut_{\mathcal{H}'}(S^*) + cut_{\mathcal{H}''}(S^*) + n^2|\{e \in \mathcal{E}''' : |e \cap S^*| = 1\}| + n^2(|S^*| - n + 1)}{|S^*|} \\ &\leq \frac{cut_{\mathcal{H}'}(S^*) + cut_{\mathcal{H}''}(S^*) + n^2(2n - |S^*|) + n^2(|S^*| - n + 1)}{|S^*|} \\ &< \frac{n|S^*|}{|S^*|} + \frac{n^2(n + 1)}{|S^*|} \\ &\leq n + n^2 \end{aligned}$$

Clearly, the density when  $|S^*| > n$  is still smaller than the density when  $|S^*| = n$ . Therefore, the size of optimal solution  $S^*$  is  $n$ .

Now, we show that for any vertex  $u \in V$ , the two copies  $u', u'' \in V^*$  satisfy either  $u' \in S^*$  and  $u'' \notin S^*$  or  $u' \notin S^*$  and  $u'' \in S^*$ . Suppose that  $u'$  and  $u''$  are both in  $S^*$ . Then for hyperedge  $e = \{u', u''\}$ ,  $f_e(|S^* \cap e|) = 0$ . By removing  $u'$  (or  $u''$ ) from  $S^*$ , we can get a larger density for set  $S^*$ , which contradicts the fact that  $S^*$  is optimal. If we assume neither  $u'$  nor  $u''$  in  $S^*$ , we can also find a contradiction by adding  $u'$  (or  $u''$ ) to  $S^*$ . Thus, either  $u'$  or  $u''$  is in  $S^*$  but not both.

With the above two properties of the optimal solution  $S^*$ , we can state that the optimal solution  $\frac{F(S^*)}{|S^*|} = n^2 + n + \frac{2 \max_{S \subseteq V} cut_{\mathcal{H}}(S)}{n}$ . Therefore, the max-cut problem can be reduced to GDSH where the input graph includes both convex and concave edge-weight functions.

## G The Network Flow Algorithm, GDSH-Flow- $\epsilon$

---

**Algorithm 3:**  $(1 - \epsilon)$  approximation algorithm for GDSH.

---

```

1 GDSH-Flow- $\epsilon(\mathcal{H})$ 
2 begin
3   Let  $e_m = \operatorname{argmax}_{e \in \mathcal{E}} f_e(|e|)$ 
4    $lb \leftarrow \frac{f_{e_m}(|e_m|)}{|e_m|}, ub \leftarrow \frac{\Psi}{1}$ 
5   while  $lb < (1 - \epsilon)ub$  do
6      $\lambda \leftarrow \sqrt{lb * ub}$ 
7     Build directed flow network  $\mathcal{G} = (U, \mathcal{A}, \lambda)$ 
8     if the cost of min-cut  $(X, Y)$  in  $\mathcal{G}$  is larger than  $\sum_{e \in \mathcal{E}} f_e(|e|)$  then
9        $ub \leftarrow \lambda$ 
10    else
11       $lb \leftarrow \lambda$ 
12    build directed flow network  $\mathcal{G} = (U, \mathcal{A}, lb)$ 
13    compute minimum cut  $(X, Y)$  from  $\mathcal{G}$ 
14  return  $X \cap V$ 

```

---

## H The Approximate Parallel Algorithm, GDSH-Para

---

**Algorithm 4:** A parallel algorithm to find the densest sub-hypergraph.

---

```

1 GDSH-Para( $\mathcal{H}$ )
2 begin
3    $S \leftarrow V, S' \leftarrow V$ 
4   for  $S \neq \emptyset$  do
5      $\Delta \leftarrow \{v \in S : F(S) - F(S \setminus \{v\}) \leq r(1 + \epsilon) \frac{F(S)}{|S|}\}$ 
6      $S \leftarrow S \setminus \Delta$ 
7     if  $\frac{F(S)}{|S|} > \frac{F(S')}{|S'|}$  then
8        $S' \leftarrow S$ 
9  return  $S'$ 

```

---

## References

1. Asahiro, Y., Iwama, K., Tamaki, H., Tokuyama, T.: Greedily finding a dense subgraph. *J. Algorithms* **34**(2), 203–221 (2000)
2. Bahmani, B., Kumar, R., Vassilvitskii, S.: Densest subgraph in streaming and mapreduce. *Proc. VLDB Endowment* **5**(5), 454–465 (2012)
3. Benson, A.R., Gleich, D.F., Leskovec, J.: Higher-order organization of complex networks. *Science* **353**(6295), 163–166 (2016)

4. Charikar, M.: Greedy approximation algorithms for finding dense components in a graph. In: Jansen, K., Khuller, S. (eds.) APPROX 2000. LNCS, vol. 1913, pp. 84–95. Springer, Heidelberg (2000). [https://doi.org/10.1007/3-540-44436-X\\_10](https://doi.org/10.1007/3-540-44436-X_10)
5. Chekuri, C., Quanrud, K., Torres, M.R.: Densest subgraph: supermodularity, iterative peeling, and flow. In: Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 1531–1555. SIAM (2022)
6. Chen, J., Saad, Y.: Dense subgraph extraction with application to community detection. *IEEE Trans. Knowl. Data Eng.* **24**(7), 1216–1230 (2010)
7. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to algorithms. MIT press (2009)
8. Dourisboure, Y., Geraci, F., Pellegrini, M.: Extraction and classification of dense communities in the web. In: Proceedings of the 16th International Conference on World Wide Web, pp. 461–470. ACM (2007)
9. Goldberg, A.V.: Finding a maximum density subgraph. Technical report, Berkeley, CA, USA (1984)
10. Goldberg, A.V., Tarjan, R.E.: A new approach to the maximum-flow problem. *J. ACM (JACM)* **35**(4), 921–940 (1988)
11. Hu, S., Wu, X., Chan, T.H.: Maintaining densest subsets efficiently in evolving hypergraphs. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, pp. 929–938. ACM (2017)
12. Kawase, Y., Miyauchi, A.: The densest subgraph problem with a convex/concave size function. *Algorithmica* **80**(12), 3461–3480 (2018)
13. Khuller, S., Saha, B.: On finding dense subgraphs. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikolettseas, S., Thomas, W. (eds.) ICALP 2009. LNCS, vol. 5555, pp. 597–608. Springer, Heidelberg (2009). [https://doi.org/10.1007/978-3-642-02927-1\\_50](https://doi.org/10.1007/978-3-642-02927-1_50)
14. Lee, Y.T., Sidford, A., Wong, S.C.: A faster cutting plane method and its implications for combinatorial and convex optimization. In: 2015 IEEE 56th Annual Symposium on Foundations of Computer Science, pp. 1049–1065. IEEE (2015)
15. Li, P., Milenkovic, O.: Inhomogeneous hypergraph clustering with applications. In: Advances in Neural Information Processing Systems, pp. 2308–2318 (2017)
16. Mitzenmacher, M., Pachocki, J., Peng, R., Tsourakakis, C., Xu, S.C.: Scalable large near-clique detection in large-scale networks via sampling. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 815–824. ACM (2015)
17. Rahman, F., Liu, W., Suhaim, S.B., Thirumuruganathan, S., Zhang, N., Das, G.: Density based clustering over location based services. In: 2017 IEEE 33rd International Conference on Data Engineering (ICDE), pp. 461–472. IEEE (2017)
18. Tsourakakis, C.: The k-clique densest subgraph problem. In: Proceedings of the 24th International Conference on World Wide Web, pp. 1122–1132. International World Wide Web Conferences Steering Committee (2015)



# Exact and Approximation Algorithms for PMMS Under Identical Constraints

Sijia Dai, Guichen Gao, Xinru Guo, and Yong Zhang<sup>(✉)</sup>

Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences,  
Shenzhen, People's Republic of China  
{sj.dai,gc.gao,xr.guo,zhangyong}@siat.ac.cn

**Abstract.** Fair division of resources is a fundamental problem in many disciplines, including computer science, economy, operations research, etc. In the context of fair allocation of indivisible goods, it is well-known that an allocation satisfying the *maximum Nash Social Welfare* (Max-NSW) is *envy-free up to one good* (EF1). In this paper, we further consider the relation between a Max-NSW allocation and two well-adopted fairness properties, i.e., *envy-free up to any good* (EFX) and *pairwise maximin share* (PMMS). In particular, we show that a Max-NSW allocation is both EFX and PMMS when agents have *identical* valuation function. Of independent interests, we also provide an algorithm for computing a PMMS allocation for *identical* variant. Moreover, we show that a  $\frac{4}{5}$ -PMMS allocation always exists and can be computed in polynomial time when agents have additive valuations and agree on the ordinal ranking of the goods (although they may disagree on the specific cardinal values).

**Keywords:** Fair division · Maximum Nash Social Welfare · Pairwise maximin share

## 1 Introduction

Fair division is an active field of work at the interface of computer science and mathematical economics. It has widespread applications in areas like vaccine distribution, kidney matching, property inheritance, government auctions, and so on. It is motivated by the realization that envy-freeness and other classic fairness notions are too demanding for the discrete setting. Alongside fairness, the efficiency of an allocation is also a desirable property. Two common measures of efficiency are that of Pareto Optimality (PO) and Nash Social Welfare [17]. In economics, Pareto Optimality is usually considered to measure the efficiency of allocation, which means no other allocation can make some agents strictly better off without making any agent strictly worse off. Mamoru and Kenjiro [17] propose the concept of Nash Social Welfare, which represents the multiplicative form of all agents' valuations w.r.t. their assigned bundles. Motivated, in part,

Supported by NSFC 12071460.

by such domains, a significant body of work in recent years has been directed towards notions of efficiency and fairness [8, 11, 12, 14–16].

The Nash Social Welfare is well-studied for divisible goods [21]. Eisenberg and Gale [13] show that the maximum Nash Social Welfare allocation can be computed in polynomial time by using the convex program. It is worth pointing out that the problem of maximizing Nash Social Welfare remains APX-hard even for additive valuations [19], i.e., there are no polynomial-time approximation schemes (PTAS) unless  $P = NP$ . Thus, it is natural to consider how to approximate the maximum Nash Social Welfare in polynomial time. Barman et al. [3] introduce the concept of Fisher market equilibrium and they design an algorithm with the approximate ratio of 1.45 and show that the output of the algorithm could achieve Pareto Optimality.

Note that there always exists a maximum Nash Social Welfare allocation, even though it is hard to find. It is interesting to investigate which kind of fairness can be achieved when an allocation reaches the maximum Nash Social Welfare. Previous studies prove that the allocation satisfying the maximum Nash Social Welfare can reach  $\pi_n$ -*Maximin Share* (MMS) (where  $\pi_n$  is a value that decreases as the number  $n$  of agents increases) and 0.618-*pairwise Maximin Share* (PMMS) [9]. An interesting family of fairness criteria has been developed around the notion of *maximin shares* which is proposed Budish [7]. MMS and PMMS mentioned above are important measurements of fairness. MMS means that everyone should at least get the value of the goods that maximizes the worst allocation of the share while PMMS considers fairness between any pair of agents. An MMS allocation may not always exist [23], but it may be approximated for some variants, e.g., if the valuations are additive, an allocation can guarantee to assign each agent a bundle with a value at least  $(\frac{3}{4} + \frac{1}{12n})$  times her maximin share [4, 6].

Envy-free (EF) is another criteria of fair division, which states that when resources are distributed among agents, every agent values her own share at least as much as she values the share of any other agent. Since an envy-free allocation cannot always be guaranteed while dividing indivisible goods, various relaxations have been studied. *Envy-free up to one good* (EF1) is proposed by Lipton et al. [20]. It is known that the maximum Nash Social Welfare allocation is also an EF1 allocation [9]. Another well-known fairness notion, *envy-free up to any good* (EFX), is introduced by Caragiannis et al. [9], which is weaker than EF but stronger than EF1. They prove that a PMMS allocation implies an EFX allocation if all agents have positive values for all goods. A good approximation of any one of EF1, EFX, MMS, and PMMS does not necessarily imply particularly strong guarantees for any of the others [1]. Most importantly, it is an open problem to resolve whether PMMS and EFX allocations always exist, even for three and four agents with additive valuation functions, respectively [10]. A reasonable approach is to focus on approximate versions of these relaxations. Georgios et al. [2] propose an efficient algorithm for an EFX allocation with the approximation ratio of 0.618 and they improve the bound to 0.717 by a modified algorithm for computing an approximation PMMS allocation. Indeed, PMMS allocations are not known to exist except in a few special cases [5, 20], such as

the PMMS allocation can be computed in polynomial time under binary additive valuations [20] and some goods are allowed to unallocated [5]. In addition, the currently best-known ratio for approximate PMMS allocation is close to 0.781 in general cases [18].

In this paper, we focus on the existence and approximation of the PMMS allocation. We investigate the properties of the maximum Nash Social Welfare allocation with the PMMS allocation and find an efficient algorithm to compute a  $\frac{4}{5}$ -PMMS allocation for the *identical ranking* variant which means that agents have additive valuations and agree on the ordinal ranking of the goods.

The remainder of this paper is structured as follows. Section 2 presents the model of fair division and some classical fairness concepts. Section 3 analyzes the relationship between the maximum Nash Social Welfare and other fairness concepts, and proves that any maximum Nash Social Welfare allocation is also PMMS allocation for any *identical* variant. Besides, we propose an algorithm to find a PMMS allocation for *identical* variant. Section 4 shows an algorithm to compute a  $\frac{4}{5}$ -PMMS allocation for the *identical ranking* variant. Section 5 gives the concluding remark and some possible future research for the fair division problem.

## 2 Preliminaries

Let  $M = \{g_1, \dots, g_m\}$  denotes the set of goods and  $N = \{1, 2, \dots, n\}$  denotes the set of agents. Throughout this paper, we assume the goods are indivisible, i.e., each good must be entirely allocated to one agent.

We consider such an allocation  $\mathbf{X} = \{X_1, \dots, X_n\}$  which is a partition of goods set  $M$ , where  $X_i \subseteq M$  is the bundle of goods assigned to agent  $i$  and  $X_i \cap X_j = \emptyset$  for any  $0 \leq i < j \leq n$ . Each agent  $i \in N$  has a valuation function  $v_i(\cdot) : 2^M \rightarrow \mathbb{R}_{\geq 0}$  that measures the utility of each agent on different subsets of goods. In this paper, we assume that the valuation function  $v_i(\cdot)$  of each agent is additive, i.e.,  $v_i(X_i) = \sum_{g \in X_i} v_i(\{g\})$  for any subset (or bundle)  $X_i$  of  $M$ . For simplicity, we write  $v_i(g)$  instead of  $v_i(\{g\})$  for good  $g \in M$ . Let us now recall some well-known notions of fairness for allocations. This gives rise to the following definition.

**Definition 1** (*Maximin Share (MMS)*).

Given  $n$  agents and a subset  $S \subseteq M$  of goods, the  $n$ -maximin share of agent  $i$  with respect to  $S$  is:

$$\mu_i(n, S) = \max_{\mathbf{X} \in \Pi_n(S)} \min_{X_j \in \mathbf{X}} v_i(X_j),$$

where  $\Pi_n(S)$  means reallocating  $M$  into  $n$  subsets  $\{X_1, \dots, X_n\}$ , which satisfies  $\cup_{i=1}^n X_i = S$  and  $\forall i, j \in N, X_i \cap X_j = \emptyset$ . When  $S = M$ , this quantity is just called the maximin share of agent  $i$ .

An allocation  $\mathbf{X}$  is an  $\alpha$ -maximin share allocation if

$$\forall i \in N, v_i(X_i) \geq \alpha \cdot \mu_i(n, M).$$

Let  $MMS_i = \max_{\mathbf{X} \in \Pi_n(M)} \min_{X_j \in \mathbf{X}} v_i(X_j)$ .



**Definition 2 (Pairwise Maximin Share (PMMS)).** An allocation  $\mathbf{X}$  is an  $\alpha$ -pairwise maximin share allocation if for any pair of agents  $i, j \in N$ ,

$$v_i(X_i) \geq \alpha \cdot \mu_i(2, X_i \cup X_j).$$

Let  $PMMS_i = \max_{j \in N \setminus \{i\}} \mu_i(2, X_i \cup X_j)$ .

**Definition 3 (PO).** An allocation  $\mathbf{X}$  is Pareto Optimal (PO) if there exist no allocation  $X'$  such that  $v_i(X'_i) \geq v_i(X_i)$  for all  $i \in N$  and  $v_j(X'_j) > v_j(X_j)$  for some  $j \in N$ .

**Definition 4 (Max-NSW).** An allocation  $\mathbf{X}^* = \{X_1^*, X_2^*, \dots, X_n^*\}$  is a maximum Nash Social Welfare (Max-NSW) allocation if it maximizes the Nash Social Welfare, i.e.,

$$\prod_{i \in N} v_i(X_i^*) \geq \prod_{i \in N} v_i(X_i)$$

for any other allocation  $\mathbf{X} = \{X_1, X_2, \dots, X_n\}$ , where  $X_i^* \cap X_j^* = \emptyset$  and  $X_i \cap X_j = \emptyset$  for any distinct  $i, j \in [n]$ .

**Definition 5 (Envy-freeness and Its Relaxations).** For any  $\alpha \in [0, 1]$ , an allocation  $\mathbf{X} = (X_i)_{i \in N}$  is:

- $\alpha$ -approximate envy-free ( $\alpha$ -EF) if

$$\forall i, j \in N, v_i(X_i) \geq \alpha \cdot v_i(X_j).$$

Note that a 1-EF allocation is an EF allocation.

- $\alpha$ -approximate envy-free up to one good ( $\alpha$ -EF1) if

$$\forall i, j \in N, \exists g \in X_j, v_i(X_i) \geq \alpha \cdot v_i(X_j \setminus g).$$

Note that a 1-EF1 allocation is an EF1 allocation.

- $\alpha$ -approximate envy-free up to any good ( $\alpha$ -EFX) if

$$\forall i, j \in N, \forall g \in X_j, v_i(X_i) \geq \alpha \cdot v_i(X_j \setminus g).$$

Note that a 1-EFX allocation is an EFX allocation.

Roughly speaking, MMS allocation focuses on the fairness among all agents, while PMMS allocations pay more attention to the fairness between pairs of agents. MMS and PMMS are both important concepts to evaluate the fairness and they are related in some particular cases [1]. However, they do not imply each other [9]. To have a clear understanding, the following example gives the MMS allocation, PMMS allocation and EFX allocation for a given set of goods respectively and shows the differences among these three kinds of allocations.

**Table 1.** Illustration on MMS, PMMS and EFX.

	$g_1$	$g_2$	$g_3$	$g_4$	$g_5$	$g_6$	$g_7$
$v_1$	1	2	2	3	3	6	4
$v_2$	5	4	9	4	3	0	6
$v_3$	0	0	1	1	9	12	2

*Example 1.* Assume that there are 3 agents and 7 goods. The valuations of agents are shown in Table 1.

Consider the allocation  $\mathbf{X}$  with  $X_1 = \{g_1, g_2, g_4, g_5, g_7\}$ ,  $X_2 = \{g_3\}$  and  $X_3 = \{g_6\}$ . We can find that  $\mathbf{X}$  is a MMS allocation, but is not a PMMS allocation or EFX allocation since  $v_2(X_2) = 9 < PMMS_2 = 15$  and  $v_2(X_2) = 9 < v_2(X_1 \setminus g_2) = 18$ .

Consider another allocation  $\mathbf{X}'$  with  $X'_1 = \{g_1, g_6\}$ ,  $X'_2 = \{g_2, g_3, g_4\}$  and  $X'_3 = \{g_5, g_7\}$ . The allocation  $\mathbf{X}'$  is a MMS allocation and PMMS allocation, but is not an EFX allocation since  $v_3(X'_3) = 11 < v_3(X'_1 \setminus g_1) = 12$ .

Consider another allocation  $\mathbf{X}''$  with  $X''_1 = \{g_6\}$ ,  $X''_2 = \{g_1, g_2, g_7\}$  and  $X''_3 = \{g_3, g_4, g_5\}$ . The allocation  $\mathbf{X}''$  is a PMMS allocation and an EFX allocation, but is not a MMS allocation since  $v_1(X''_1) = 6 < MMS_1 = 7$ .

In this paper, we consider PMMS allocations for the following variants.

- *Identical:* For any  $g_j \in M$ , each agent has the same valuation. Without loss of generality, we may use  $v(g_j)$  to denote the value of good  $g_j$  for all agents;
- *Identical Ranking:* Agents have additive valuations and agree on the ordinal ranking of the goods.

### 3 The PMMS Allocation for Identical Variant

Existing works indicate that Max-NSW allocation can also reach 0.618-PMMS and 0.618-EFX [9]. The bound of PMMS is proved to be tight, while whether the bound of EFX is best possible is still open. In this section, we show that PMMS and EFX can be achieved if all agents' valuation functions are *identical*.

**Theorem 1.** *A Max-NSW allocation  $\mathbf{X}^*$  is PMMS and EFX if the valuation functions are identical.*

*Proof.* We prove this theorem by contradiction. If a Max-NSW allocation  $\mathbf{X}^*$  is not PMMS, there must exist two agents  $i$  and  $j$ , such that

$$v(X_i^*) < \max_{B \in \Pi_2(S)} \min \{v(B_1), v(B_2)\}.$$

Reallocate the set  $S = X_i^* \cup X_j^*$ , which allocates to agents  $i$  and  $j$ , into  $B_1$  and  $B_2$ , where  $B_1 = \arg \max_{B \in \Pi_2(S)} \min \{v(B_1), v(B_2)\}$  and  $B_2 = S \setminus B_1$ .

Construct a new allocation  $X'$  such that  $X'_i = B_1$ ,  $X'_j = B_2$  and  $X'_k = X_k^*$  for all  $k \neq i, j$ . We have

$$v(X'_i) = \max_{B \in \Pi_2(S)} \min \{v(B_1), v(B_2)\} \leq \frac{v(X_i^*) + v(X_j^*)}{2}.$$

Since  $v(X_i^*) + v(X_j^*) = v(X'_i) + v(X'_j)$ ,

$$v(X_j^*) - v(X'_j) = v(X'_i) - v(X_i^*) \leq \frac{v(X_j^*) - v(X_i^*)}{2}. \tag{1}$$

Therefore,

$$\begin{aligned} & v(X'_i) \cdot v(X'_j) \\ &= (v(X_i^*) + v(X'_i) - v(X_i^*)) \cdot (v(X_j^*) - (v(X'_i) - v(X_i^*))) \\ &= v(X_i^*) \cdot v(X_j^*) + (v(X'_i) - v(X_i^*)) \cdot (v(X_j^*) - v(X'_i)) \\ &> v(X_i^*) \cdot v(X_j^*). \end{aligned}$$

The last inequality strictly holds since (i)  $v(X'_i) > v(X_i^*)$  by the assumption that  $X^*$  is not PMMS and (ii)  $v(X_j^*) - v(X'_i) \geq v(X'_i) - v(X_i^*) > 0$  according to Inequality (1).

Since  $v_k(X'_k) = v_k(X_k^*)$  for all  $k \neq i, j$ , it follows that

$$\prod_{\ell \in N} v(X'_\ell) > \prod_{\ell \in N} v(X_\ell^*)$$

which contradicts the statement that  $\mathbf{X}^*$  is Max-NSW.

Now we know that a Max-NSW allocation  $\mathbf{X}^*$  is PMMS if the valuation functions are *identical*. Combined with the fact that PMMS allocation implies EFX allocation when  $v(g) > 0$  for all  $g \in M$  [9], Theorem 1 is correct.  $\square$

Note that PMMS implies EFX, but EFX does not imply PMMS [9]. According to the following example, we can see that PMMS is stronger than EFX even when the valuation functions are *identical*. Thus, computing PMMS allocation could not use **Algorithm 6.1** [22] to find EFX allocation for *identical* variants.

*Example 2.* Assume there are two agents 1, 2 and 5 goods. The agents have identical valuation functions, as shown in Table 2.

**Table 2.** Illustration on identical valuation function.

	$g_1$	$g_2$	$g_3$	$g_4$	$g_5$	$g_6$
$v$	8	4	3	3	2	2

Consider the allocation  $\mathbf{X}$  that is computed by **Algorithm 6.1** [22] with  $X_1 = \{g_1, g_5\}$ ,  $X_2 = \{g_2, g_3, g_4, g_6\}$ . It can be seen that  $\mathbf{X}$  is an EFX allocation, but is not a PMMS allocation since

---

**Algorithm 1:** PMMS-cycle Elimination

---

**1 Input:** Sets of agents  $N$ , set of goods  $M$ , any complete allocation  $\mathbf{X} = \{X_1, X_2, \dots, X_n\}$ ;  
**2 Output:** a complete allocation  $\mathbf{X}'$ ;  
**3** Construct the *PMMS Graph*  $G_{\mathbf{X}}$ ;  
**4 while**  $G_{\mathbf{X}}$  contains a cycle  $\mathcal{C}: c_1 \rightarrow c_2 \rightarrow \dots \rightarrow c_r \rightarrow c_1$  **do**  
**5**     **for**  $k = 1$  to  $r - 1$  **do**  
**6**          $X'_{c_k} \leftarrow X_{c_{k+1}}$ ;  
**7**          $X'_{c_r} \leftarrow X_{c_1}$ ;  
**8**         Update  $G_{\mathbf{X}}$ ;  
**return**  $\mathbf{X}'$ ;  


---

$$v_1(X_1) = 10 < PMMS_1 = \max_{B \in \prod_2(X_1 \cup X_2)} \min \{v_1(B_1), v_1(B_2)\} = 11.$$

Consider another allocation  $\mathbf{X}'$  with  $X'_1 = \{g_1, g_3\}$  and  $X'_2 = \{g_2, g_4, g_5, g_6\}$ . It can be verified that  $\mathbf{X}'$  satisfies both PMMS and EFX.

According to the above analysis, a PMMS allocation exists in the *identical* variant, but there is no good method to find it. In the following part of this section, we introduce an interesting structure, *PMMS Graph*. With the help of such a structure, a novel algorithm is proposed to obtain the PMMS allocation for the *identical* variant.

*PMMS Graph*  $G_{\mathbf{X}} = (V_{\mathbf{X}}, E_{\mathbf{X}})$  is a directed graph and defined for an allocation  $\mathbf{X}$ . Each vertex  $v \in V_{\mathbf{X}}$  corresponds to an agent  $i \in N$ . There is a directed edge  $(i, j) \in E_{\mathbf{X}}$  if and only if

$$v_i(X_i) < \max_{j \in N \setminus \{i\}} \mu_i(2, X_i \cup X_j).$$

Note that the right part is  $PMMS_i$ . If  $\exists k \in N \setminus \{i, j\}$  such that

$$\mu_i(2, X_i \cup X_k) = PMMS_i,$$

and  $v_i(X_i) < PMMS_i$ , we may choose either  $(i, j)$  or  $(i, k)$  to be an edge.

An allocation is *complete* if every good  $g \in M$  is assigned to some agent in  $N$ .

In Algorithm 1, we construct the *PMMS Graph* for any complete allocation  $\mathbf{X}$  firstly, and then if there exists a cycle in the Graph, we exchange only the entire bundle of agents, not goods. Besides, Algorithm 1 can be implemented in polynomial time.

**Lemma 1.** *The PMMS Graph is acyclic after running Algorithm 1.*

---

**Algorithm 2:** Finding a PMMS allocation for *identical* variant

---

- 1 **Input:** Sets of agents  $N$ , set of goods  $M$ , any complete allocation  $\mathbf{X} = \{X_1, X_2, \dots, X_n\}$ ;
- 2 **Output:** a Complete PMMS allocation  $\mathbf{X}'$ ;
- 3 Construct the *PMMS Graph*  $G_{\mathbf{X}}$ ;
- 4 **while** there is a vertex  $j$  with in-degree  $> 0$  in  $G_{\mathbf{X}}$  and  $v(X_i)$  is maximized among all such  $j$ s **do**
- 5     Choose the vertex  $i$  such that  $(i, j) \in E_{\mathbf{X}}$  and  $v(X_i)$  is minimized among all such  $i$ s;
- 6      $S_2 \leftarrow \arg \max_{B \in \Pi_2 X_i \cup X_j} \min \{v(B_1), v(B_2)\}$ ;
- 7      $S_1 \leftarrow (X_i \cup X_j) \setminus S_2$ ;
- 8      $\mathbf{X}' \leftarrow \mathbf{X}$ ;
- 9      $X'_i \leftarrow S_2$ ;
- 10     $X'_j \leftarrow S_1$ ;
- 11    Update  $G_{\mathbf{X}}$  according to the current allocation;

---

**Return**  $\mathbf{X}'$ ;

---

*Proof.* Assuming that there is a cycle  $\mathcal{C} = \{c_1, c_2, \dots, c_r\}$  in  $G_{\mathbf{X}}$ , consider the cycle  $c_1 \rightarrow c_2 \rightarrow \dots \rightarrow c_r \rightarrow c_1$ . Cycle  $\mathcal{C}$  can be eliminated by reallocating bundles. Precisely speaking, if  $\mathbf{X} = \{X_1, \dots, X_n\}$ , we can obtain  $\mathbf{X}' = \{X'_1, \dots, X'_n\}$  by reallocating the goods as follows:  $X'_i = X_i$  for all  $i \notin \{c_1, c_2, \dots, c_r\}$ , and  $X'_{c_1} = X_{c_2}, X'_{c_2} = X_{c_3}, \dots, X'_{c_r} = X_{c_1}$ .

After each round, the valuations of all agents in cycle  $\mathcal{C}$  are strictly increased. Note that  $\mathbf{X}'$  uses the same bundles of goods as  $\mathbf{X}$ , they are only assigned to different agents. Thus, each vertex will be updated in at most  $n$  rounds and finally, the *PMMS Graph* after running Algorithm 1 is acyclic.  $\square$

For any allocation, we can construct a *PMMS Graph* accordingly. If a *PMMS Graph* is in corresponding to the *identical* variant, from the definition of *PMMS Graph*, there is no directed path with a length longer than 1.

To find the PMMS allocation in *identical* variant, we start from any allocation. At each step, the goods allocated to the most envied agent, say  $i$ , and the agent who envies  $i$  most will be reallocated to balance their valuations as much as possible. Roughly speaking, each loop in Algorithm 1 monotonously decreases the envy of the allocation. Such processing can be implemented in the *PMMS Graph*.

**Corollary 1.** *When the valuations are identical, a PMMS allocation can be achieved via Algorithm 2.*

*Proof.* In each step, the update will strictly decrease a higher valuation or strictly increase a lower valuation. Since the valuations among all agents are not infinity, the algorithm must be terminated and leads to an empty *PMMS Graph* with no edge.  $\square$

---

**Algorithm 3:** Find an EFX allocation for *identical rankings*

---

```

1 Input: Sets of agents  $N$ , set of goods  $M$ ;
2 Output: An allocation  $\mathbf{X}$ ;
3 Order the goods in descending order of value, i.e.,
4  $v(g_1) \geq v(g_2) \geq \dots \geq v(g_m) > 0$ ;
5 Set  $\mathbf{X} \leftarrow (\emptyset, \emptyset, \dots, \emptyset)$ ;
6 Construct the envy graph  $G_{\mathbf{X}}$ ;
7 for  $\ell = 1$  to  $m$  do
8   while there is no node of in-degree 0 in  $G_{\mathbf{X}}$  do  $\triangleright$  EliminateEnvyCycles
9     Find a cycle  $\mathcal{C}: c_1 \rightarrow c_2 \rightarrow \dots \rightarrow c_r \rightarrow c_1$  in  $G_{\mathbf{X}}$ ;
10    for  $k = 1$  to  $r - 1$  do
11       $X'_{c_k} \leftarrow X_{c_{k+1}}$ ;
12     $X'_{c_r} \leftarrow X_{c_1}$ ;
13    Update  $G_{\mathbf{X}}$ ;
14  Set  $j \in N$  be a node of in-degree 0;  $\triangleright$  ties are broken lexicographically
15   $X_j \leftarrow X_j \cup \{g_\ell\}$ ;
16  Update  $G_{\mathbf{X}}$ ;
Return  $\mathbf{X}$ ;

```

---

Note that the PMMS allocation for *identical* variant can be reduced from the 2-partition problem when the number of agents is 2, so computing PMMS allocation is NP-hard.

#### 4 A $\frac{4}{5}$ -PMMS Allocation for Identical Ranking

We analyze **Algorithm 6.1** that is provided for computing the EFX allocation in *identical ranking* variant [22]. We also prove that it guarantees a  $\frac{4}{5}$ -PMMS allocation. Algorithm 3 is rewritten on the basis of **Algorithm 6.1**.

In order to modify the *envy-cycle-elimination algorithm* (Algorithm 3) of Lipton et al. [20]. We first need to introduce the notion of an *envy graph*. *Envy Graph*  $G_{\mathbf{X}} = (V_{\mathbf{X}}, E_{\mathbf{X}})$  is a directed graph and defined for an allocation  $\mathbf{X}$ . Each vertex  $v \in V_{\mathbf{X}}$  corresponds to an agent  $i \in N$ . There is a directed edge  $(i, j) \in E_{\mathbf{X}}$  if and only if

$$v_i(X_i) < v_i(X_j).$$

Our algorithm (Algorithm 3) allocates the goods one by one in descending order of their value. In each iteration, a good is assigned to the agent who is not envied. If there is no agent, we do the *envy-cycle-elimination algorithm* to break the envy cycle until such an agent is found.

**Theorem 2.** *For additive valuations with identical rankings, Algorithm 3 terminates with a  $\frac{4}{5}$ -PMMS allocation in  $O(mn^3) + O(m \log m)$  time.*

*Proof.* Let  $X^\ell$  be the allocation maintained by Algorithm 3 at the end of the  $\ell^{th}$  iteration. It suffices to show that for each  $\ell \in [m]$ , if  $X^\ell$  is a  $\frac{4}{5}$ -PMMS allocation, then so is  $X^{\ell+1}$ .

We first argue that at all times, any pair of agents  $i, j$  satisfies EFX property according to **Theorem 6.2** [22]. It means that  $\forall i, j \in N, v_i(X_i^\ell) \geq v_i(X_j^\ell \setminus g_\ell)$  at each round  $\ell$ , where  $g_\ell$  is the good most recently added to what is currently  $X_j$ .

Recall that Algorithm 3 assigns the good  $g_\ell$  to agent  $j$  in the  $\ell^{th}$  iteration, i.e., thus  $X_j^\ell = X_j^{\ell-1} \cup \{g_\ell\}$ . At the same time, the allocation of any other  $i \in N \setminus \{j\}$  is unchanged before the *EliminateEnvyCycles* operation. Therefore, in order to establish that  $X^\ell$  is a  $\frac{4}{5}$ -PMMS allocation, we only need to consider agent  $j$  and show that  $v_i(X_i^\ell) \geq \frac{4}{5} \cdot \mu_i(2, X_i^\ell \cup X_j^\ell)$  for all  $i \in N \setminus \{j\}$ .

Since Algorithm 3 processes the goods in decreasing order of value, the good  $g_\ell$  is the least valued good in  $X_j^\ell$ . Due to agent  $j$  gets the good at the end of the  $\ell^{th}$  iteration, agent  $j$  will be satisfied  $\frac{4}{5}$ -PMMS property. Firstly, we need to analyze for all  $i \in N \setminus \{j\}$  satisfies  $\frac{4}{5}$ -PMMS property.

We analyze two cases,  $|X_i^\ell| = 1$  and  $|X_i^\ell| \geq 2$ , separately.

**Case 1.**  $|X_i^\ell| = 1$ .

In this case, agent  $i$  has one good only. At the same time,  $v_i(X_i^\ell) \geq v_i(X_j^{\ell-1}) = v_i(X_j^\ell) - v_i(g_\ell)$  is satisfied according to the agent selection rule of Algorithm 3. Thus, agent  $i$  must satisfy PMMS property.

**Case 2.**  $|X_i^\ell| \geq 2$ .

Due to  $|X_i^\ell| \geq 2$ , we have  $v_i(X_i^\ell) \geq 2 \cdot v_i(g_\ell)$  obviously. We can find that the following inequality is satisfied.

$$\begin{aligned} v_i(X_i^\ell) &= \frac{4}{5} \cdot v_i(X_i^\ell) + \frac{1}{5} \cdot v_i(X_i^\ell) \\ &\geq \frac{4}{5} \cdot v_i(X_i^\ell) + \frac{2}{5} \cdot v_i(g_\ell) = \frac{4}{5} \cdot (v_i(X_i^\ell) + \frac{1}{2} \cdot v_i(g_\ell)) \\ &= \frac{4}{5} \cdot \frac{v_i(X_i^\ell) + v_i(X_i^\ell) + v_i(g_\ell)}{2} \geq \frac{4}{5} \cdot \frac{v_i(X_i^\ell) + v_i(X_j^\ell)}{2} \\ &\geq \frac{4}{5} \cdot \mu_i(2, X_i^\ell \cup X_j^\ell) \end{aligned}$$

Here, the inequality holds by  $|X_i^\ell| \geq 2$  and EFX property.

Then we proof that the agent  $j$  who gets the good in  $\ell^{th}$  iteration will satisfies  $\frac{4}{5}$ -PMMS property. When some other agent  $k \in N \setminus \{j\}$  gets the last good  $g'$  before  $\ell^{th}$  iteration, agent  $j$  satisfy

$$\begin{aligned} v_j(X_j^\ell) &= v_j(X_j^{\ell-1}) + v_j(g_\ell) \geq \frac{4}{5} \cdot \frac{v_j(X_k^\ell) + v_j(X_j^{\ell-1})}{2} + v_j(g_\ell) \\ &> \frac{4}{5} \cdot \frac{v_j(X_k^\ell) + v_j(X_j^{\ell-1}) + v_j(g_\ell)}{2} \geq \frac{4}{5} \cdot \mu_j(2, X_k^\ell \cup X_j^\ell). \end{aligned}$$

Combining the proof above, Algorithm 3 terminates with a  $\frac{4}{5}$ -PMMS allocation. Finally, we show that Algorithm 3 terminates in  $O(mn^3) + O(m \log m)$

time. Our algorithm involves a single sorting step and at most  $O(n^3)$  *EliminateEnvyCycles* operation for each good, thus it requires  $O(mn^3) + O(m \log m)$  time overall.  $\square$

## 5 Concluding Remarks

We study the relationship between the maximum Nash Social Welfare and fairness concepts. It is shown that a maximum Nash Social Welfare allocation is always PMMS and EFX if the valuation functions of agents are identical. Moreover, we established the universal existence of *pairwise maximin share* allocations for *identical* variants. *PMMS graph* is introduced to help us find the PMMS allocation for the identical variant. This structure might give us a hint to find the PMMS allocation if some properties are satisfied. Additionally, we proved the efficient computability of approximation PMMS allocations.

Fair allocation is fundamental and some basic concepts have been well studied during these years. However, some important issues are still not clear. E.g., whether EFX allocation exist when the number of agents is greater than 3? Whether PMMS allocation always exists? Establishing the existence of complete PMMS allocations under general valuations would also be interesting. Furthermore, is it possible to find an approximately PMMS allocation which approximation ratio of more than 0.781 for general variant?

## References

1. Amanatidis, G., Birmpas, G., Markakis, V.: Comparing approximate relaxations of envy-freeness. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence, pp. 42–48. IJCAI2018, AAAI Press, Stockholm, Sweden (2018)
2. Amanatidis, G., Markakis, E., Ntokos, A.: Multiple birds with one stone: beating  $1/2$  for EFX and GMMS via envy cycle elimination. *Theoret. Comput. Sci.* **841**, 94–109 (2020)
3. Barman, S., Krishnamurthy, S.K., Vaish, R.: Finding fair and efficient allocations. In: Proceedings of the 19th ACM Conference on Economics and Computation, pp. 557–574. EC 2018, ACM, New York, NY, USA (2018)
4. Barman, S., Murthy, S.K.K.: Approximation algorithms for maximin fair division. In: Proceedings of the 18th ACM Conference on Economics and Computation, pp. 647–664. EC 2017, ACM, New York, NY, USA (2017)
5. Barman, S., Verma, P.: Existence and computation of maximin fair allocations under matroid-rank valuations. In: Dignum, F., Lomuscio, A., Endriss, U., Nowé, A. (eds.) Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems, pp. 169–177. AAMAS2021, ACM, Virtual Event, United Kingdom (2021)
6. Bouveret, S., Lemaître, M.: Characterizing conflicts in fair division of indivisible goods using a scale of criteria. *Auton. Agents Multi Agent Syst.* **30**(2), 259–290 (2016)
7. Budish, E.: The combinatorial assignment problem: approximate competitive equilibrium from equal incomes. *J. Polit. Econ.* **119**(6), 1061–1103 (2011)



8. Cai, Z., Zheng, X.: A private and efficient mechanism for data uploading in smart cyber-physical systems. *IEEE Trans. Netw. Sci. Eng.* **7**(2), 766–775 (2020)
9. Caragiannis, I., Kurokawa, D., Moulin, H., Procaccia, A.D., Shah, N., Wang, J.: The unreasonable fairness of maximum Nash welfare. *ACM Trans. Econ. Comput.* **7**(3), 1–32 (2019)
10. Chaudhury, B.R., Garg, J., Mehlhorn, K.: EFX exists for three agents. In: *Proceedings of the 21st ACM Conference on Economics and Computation*, pp. 1–19. EC 2020, ACM, New York, NY, USA (2020)
11. Chin, F.Y.L., et al.: Competitive algorithms for unbounded one-way trading. *Theor. Comput. Sci.* **607**, 35–48 (2015)
12. Dolev, D., Feitelson, D.G., Halpern, J.Y., Kupferman, R., Linial, N.: No justified complaints: on fair sharing of multiple resources. In: *Innovations in Theoretical Computer Science*, pp. 68–75. ITCS2012, ACM, New York, NY, USA (2012)
13. Eisenberg, E., Gale, D.: Consensus of subjective probabilities: the pari-mutuel method. *Ann. Math. Stat.* **30**(1), 165–168 (1959)
14. Etsion, Y., Ben-Nun, T., Feitelson, D.G.: A global scheduling framework for virtualization environments. In: *23rd IEEE International Symposium on Parallel and Distributed Processing*, pp. 1–8. IPDPS 2009, IEEE, Rome, Italy (2009)
15. Ghodsi, A., Zaharia, M., Hindman, B., Konwinski, A., Shenker, S., Stoica, I.: Dominant resource fairness: Fair allocation of multiple resource types. In: *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation*, pp. 323–336. NSDI2011, USENIX Association, USA (2011)
16. Grandl, R., Ananthanarayanan, G., Kandula, S., Rao, S., Akella, A.: Multi-resource packing for cluster schedulers. In: *ACM SIGCOMM 2014 Conference*, pp. 455–466. SIGCOMM 2014, ACM, Chicago, IL, USA (2014)
17. Kaneko, M., Nakamura, K.: The Nash social welfare function. *Econometrica* **47**(2), 423–435 (1979)
18. Kurokawa, D.: *Fair Division in Game Theoretic Settings*. Carnegie Mellon University (2017)
19. Lee, E.: APX-hardness of maximizing Nash social welfare with indivisible items. *Inf. Process. Lett.* **122**, 17–20 (2017)
20. Lipton, R., Markakis, E., Mossel, E., Saberi, A.: On approximately fair allocations of indivisible goods. In: *Proceedings of the 5th ACM Conference on Electronic Commerce*, pp. 125–131. EC 2004, ACM, New York, NY, USA (2004)
21. Moulin, H.: *Fair division and collective welfare*, vol. 1. MIT Press, 1st edn. (2003)
22. Plaut, B., Roughgarden, T.: Almost envy-freeness with general valuations. *SIAM J. Discret. Math.* **34**(2), 1039–1068 (2020)
23. Procaccia, A.D., Wang, J.: Fair enough: guaranteeing approximate maximin shares. In: *Proceedings of the 15th ACM Conference on Economics and Computation*, pp. 675–692. EC 2014, ACM, New York, NY, USA (2014)



# Finite-State Relative Dimension, Dimensions of AP Subsequences and a Finite-State van Lambalgen's Theorem

Satyadev Nandakumar<sup>✉</sup>, Subin Pulari<sup>✉</sup>, and Akhil S<sup>✉</sup>

Department of Computer Science and Engineering, Indian Institute of Technology  
Kanpur, Kanpur, U.P, India  
{satyadev,subinp,akhis}@cse.iitk.ac.in

**Abstract.** Finite-state dimension (Dai, Lathrop, Lutz, and Mayordomo (2004)) quantifies the information rate in an infinite sequence as measured by finite-state automata. In this paper, we define a relative version of finite-state dimension. The finite-state relative dimension  $\dim_{FS}^Y(X)$  of a sequence  $X$  relative to  $Y$  is the finite-state dimension of  $X$  measured using the class of finite-state gamblers with an oracle access to  $Y$ . We show its mathematical robustness by equivalently characterizing this notion using the relative block entropy rate of  $X$  conditioned on  $Y$ .

We derive inequalities relating the dimension of a sequence to the relative dimension of its subsequences along any arithmetic progression (A.P.). These enable us to obtain a strengthening of Wall's Theorem on the normality of A.P. subsequences of a normal number, in terms of relative dimension. In contrast to the original theorem, this stronger version has an exact converse yielding a new characterization of normality.

We also obtain finite-state analogues of van Lambalgen's theorem on the symmetry of relative normality.

**Keywords:** Finite-state relative dimension · Wall's theorem on the normality of AP subsequences · van Lambalgen's theorem

## 1 Introduction

Finite-state dimension, introduced by Dai, Lathrop, Lutz and Mayordomo [7] measures the information density in a sequence as measured by finite-state betting algorithms called finite-state  $s$ -gales. Every sequence drawn from a finite alphabet has a well-defined finite-state dimension between 0 and 1. We can view dimension 1 sequences as being “finite-state random”, and dimension 0 sequences as being easily predictable using finite-state  $s$ -gales. This notion was shown to be mathematically robust, having several equivalent characterizations - using finite-state gamblers [7], block entropy rates, and finite-state compressibility ratios [3]. Due to this equivalence with finite-state compressibility, a result due to Schnorr

and Stimm [12] implies that a sequence is Borel normal if and only if it has finite-state dimension 1, establishing connections to metric number theory and probability.

In this work, we introduce a notion of the *relative* finite-state dimension of a sequence  $X$  with respect to any arbitrary sequence  $Y$ . Intuitively, this is the information density in  $X$  as viewed by a finite-state machine which has “oracle” access to bits from  $Y$ . We show that this leads to a meaningful notion of relative finite-state dimension based on finite-state  $s$ -gales with oracle access. We provide an equivalent characterization of this notion using *conditional* block entropy rates, generalizing the insight in the work of Bourke, Hitchcock and Vindochandran [3]. This demonstrates the mathematical robustness of our definition. Our notion is a finite-state analogue of conditional entropy, in a similar manner as mutual dimension introduced by Case and Lutz [4] is an analogue of mutual information.

We use our notion of relative finite-state dimension to derive a stronger version of Wall’s theorem on subsequences along arithmetic progressions (A.P.) of normal sequences. While Wall’s theorem states that A.P. subsequences of any normal sequence  $X$  must be normal, we show that in addition, such an A.P. subsequence must also be relatively normal with respect to all other A.P. subsequences of  $X$  with the same common difference. This version has a precise converse, unlike Wall’s original theorem. This yields a new *characterization* of normality in terms of its A.P. subsequences.

We finally study the symmetry of relative finite-state randomness, showing that  $X$  is finite-state relatively random to  $Y$  if and only if the converse holds. This can be viewed as a finite-state version of van Lambalgen’s theorem [10] establishing that relative Martin-Löf randomness is symmetric. Since van Lambalgen’s theorem analogues fail to hold for other randomness notions [1, 5, 8, 16], this shows that finite-state randomness is a rare setting where symmetry of relative randomness does hold. We show, further that this result does not generalize to arbitrary finite-state dimensions.

As a consequence of this analogue of van Lambalgen’s theorem, we establish that finite-state independence, introduced by Becher, Carton and Heiber [2], implies relative finite-state normality. The converse direction remains open.

## 2 Preliminaries

We consider a finite alphabet  $\Sigma$ . The set of finite strings from  $\Sigma$  is denoted as  $\Sigma^*$  and the set of infinite sequences, by  $\Sigma^\infty$ . For any finite string  $x$ , we denote its length by  $|x|$ . We use the notation  $x[i : j]$ , where  $0 \leq i < j < |x|$ , to denote the substring of  $x$  from position  $i$  to  $j$ , both ends inclusive. The character at position  $n$  is denoted by  $x[n]$ . We use the notation  $x[: n]$  to denote  $x[0 : n]$ . We use similar notations for infinite sequences. For  $\ell \in \mathbb{N}$ ,  $\Sigma^{<\ell}$  denotes the set of strings of length less than  $\ell$ . Logarithms in this work have base 2. For any  $n \in \mathbb{N}$ ,  $[n]$  denotes the set  $\{0, 1, \dots, n - 1\}$ . Given infinite sequences  $Y_0, Y_1, \dots, Y_{n-1}$ , we define the *product sequence*  $Y_0 \times Y_1 \times Y_2 \times \dots \times Y_{n-1}$  to be the infinite sequence

whose  $i^{\text{th}}$  digit is  $(Y_0[i], Y_1[i], Y_2[i], \dots, Y_{n-1}[i])$ . And, we define the *interleaved sequence*  $Y_0 \oplus Y_1 \oplus Y_2 \oplus \dots \oplus Y_{n-1}$  to be the sequence in  $\Sigma^\infty$  whose  $i^{\text{th}}$  bit is equal to  $Y_{i \bmod n}[\lfloor i/n \rfloor]$ . For any natural number  $n > 0$ ,  $\text{Bij}(n)$  denotes the set of permutations on  $[n]$ .

### 3 Finite-State Relative Dimension

Finite-state dimension, introduced by Dai, Lathrop, Lutz and Mayordomo [7], characterizes the information density of an infinite sequence as perceived by a finite-state automata. In this work, we introduce the idea of finite-state relative dimension of an infinite sequence  $X$  with respect to another an infinite sequence  $Y$ . We can interpret this notion as the finite-state asymptotic information density of  $X$  relative to  $Y$ .

We formulate finite-state relative dimension using an extension of the finite-state gambler (see [7]) which we call as finite state relative gambler. In contrast to the classic finite-state gambler, the finite-state relative gambler betting on an infinite sequence  $X$  is given access to the characters of another infinite sequence  $Y$ . The gambler may utilize the information obtained from the characters of  $Y$  while betting on  $X$ . To preserve the finite nature of the model, the gambler is allowed access to only a finite window of characters of  $Y$ . This finite window shifts forward in  $Y$ , in a sliding manner as the gambler processes  $X$ .

We assume that  $X$  and  $Y$  are elements from  $\Sigma_1^\infty$  and  $\Sigma_2^\infty$  respectively.

**Definition 1.** *A finite-state relative gambler (FSRG) with window length  $k$  is a 5-tuple  $G = (Q, \delta, \beta, q_0, c_0)$  where:*

- \*  $Q$  is a nonempty, finite set of states,
- \*  $\delta : Q \times \Sigma_2^k \times \Sigma_1 \rightarrow Q$  is the transition function,
- \*  $\beta : Q \times \Sigma_2^k \times \Sigma_1 \rightarrow \mathbb{Q} \cap [0, 1]$  is the betting function,  
 such that  $\forall q \in Q, y \in \Sigma_2^k, \sum_{x \in \Sigma_1} \beta(q, y, x) = 1$ ,
- \*  $q_0 \in Q$  is the initial state,
- \*  $c_0$  is the initial capital, generally set to 1.

Let a FSRG  $G$  be in state  $q \in Q$ . After reading the oracle word  $y \in \Sigma_2^k$ , the gambler places the bet  $\beta(q, y, x)$  on the next character  $x \in \Sigma_1$  of  $X$ . After processing  $x$ , the input pointer to  $X$  as well as  $Y$  move forward by a single position.  $G$  enters the state  $\delta(q, y, x)$ .

Analogous to [7], we define the concept of an  $s$ -gale induced by a FSRG and the corresponding success sets.

**Definition 2.** *( $s$ -gale induced by a FSRG) Let  $G = (Q, \delta, \beta, q_0, c_0)$  be a FSRG with window length  $k$ . For an  $s \in [0, \infty)$ , the  $s$ -gale of  $G$  with oracle access to  $Y$  is the function  $d_{G,Y}^{(s)} : \Sigma_1^* \rightarrow [0, \infty)$  defined by  $d_{G,Y}^{(s)}(\lambda) = c_0$  and for every  $w \in \Sigma^*$  and  $x \in \Sigma$ , by*

$$d_{G,Y}^{(s)}(wx) = |\Sigma_1|^s \times \beta(q, Y[\lceil |w| : |w| + k - 1 \rceil], x) \times d_{G,Y}^{(s)}(w).$$

We call an  $s$ -gale induced by a finite-state relative gambler as a finite-state relative  $s$ -gale.

When the automata is in state  $q$ , after reading the oracle word  $y \in \Sigma_2^k$  and the next character  $x \in \Sigma_1$ , the  $s$ -gale function gets multiplied by a factor of  $|\Sigma_1|^s \times \beta(q, y, x)$ . As remarked in [7], this corresponds to “betting under inflation”, since the expected value after any bet is less than that before the bet.

**Definition 3.** (*Success set of a finite-state relative  $s$ -gale*) We say that the finite-state relative  $s$ -gale  $d_{G,Y}^{(s)}$  succeeds on a sequence  $X \in \Sigma_1^\infty$  if  $\limsup_{n \rightarrow \infty} d_{G,Y}^{(s)}(X[:n]) = \infty$ . The success set of  $d_{G,Y}^{(s)}$  is

$$S^\infty[d_{G,Y}^{(s)}] = \{X \in \Sigma_1^\infty \mid d_{G,Y}^{(s)} \text{ succeeds on } X\}.$$

For any  $X \in \Sigma_1^\infty$  and  $Y \in \Sigma_2^\infty$ ,  $\mathcal{G}^Y(X)$  is the set of all  $s \in [0, \infty)$  such that there is a finite-state relative  $s$ -gale  $d_{G,Y}^{(s)}$  for which  $X \in S^\infty[d_{G,Y}^{(s)}]$ .

**Definition 4.** (*Finite-state relative dimension*) The finite-state relative dimension of a sequence  $X \in \Sigma_1^\infty$  with respect to a sequence  $Y \in \Sigma_2^\infty$  is

$$\dim_{FS}^Y(X) = \inf \mathcal{G}^Y(X).$$

## 4 Relative Block Entropy Rates and Finite-State Relative Dimension

Bourke, Hitchcock and Vinodchandran [3] characterize finite-state dimension using disjoint block entropy rates. We formulate the notion of relative disjoint block entropy rates, extending the definitions in [3]. We characterize finite-state relative dimension using relative block entropy rates.

### 4.1 Relative Block Entropy Rates

Let  $n, \ell \in \mathbb{N}$ , where  $n$  divides  $l$ . Given a string  $Y \in \Sigma_2^n$  and  $y \in \Sigma_2^\ell$ , let

$$N(y, Y) = |\{0 \leq i \leq n/\ell \mid Y[i\ell : (i+1)\ell - 1] = y\}|$$

be the number of times  $y$  occurs in  $\ell$  length blocks of  $Y$ . The *disjoint block frequency* of  $y$  in  $Y$  is defined as

$$P(y, Y) = \frac{\ell}{n} N(y, Y).$$

Given  $X \in \Sigma_1^n$  and  $Y \in \Sigma_2^n$  and  $x \in \Sigma_1^\ell$  and  $y \in \Sigma_2^\ell$ , let

$$N(x, X; y, Y) = \left| \left\{ 0 \leq i \leq \frac{n}{\ell} \mid X[i\ell : (i+1)\ell - 1] = x \wedge Y[i\ell : (i+1)\ell - 1] = y \right\} \right|.$$

This is the number of times  $x$  and  $y$  occur concurrently in the  $\ell$  length blocks of  $X$  and  $Y$  respectively. Then the *disjoint block frequency* of  $x$  in  $X$  relative to  $y$  in  $Y$  is defined as

$$P(x, X | y, Y) = \frac{N(x, X; y, Y)}{N(y, Y)}$$

It is easy to see that for any  $y \in \Sigma_2^\ell$ ,  $\sum_{x \in \Sigma_1^\ell} P(x, X | y, Y) = 1$ . For two strings  $X \in \Sigma_1^n$  and  $Y \in \Sigma_2^n$ , the  $\ell$ -length block entropy of  $X$  relative to  $Y$  is defined as:

$$H_\ell(X|Y) = \frac{1}{\ell \log(|\Sigma_1|)} \sum_{y \in \Sigma_2^\ell} P(y, Y) \sum_{x \in \Sigma_1^\ell} -P(x, X | y, Y) \log(P(x, X | y, Y)).$$

**Definition 5.** For  $X \in \Sigma_1^\infty$  and  $Y \in \Sigma_2^\infty$ , the block entropy rate of  $X$  relative to  $Y$  is defined as:

$$H(X|Y) = \inf_{\ell \in \mathbb{N}} \liminf_{k \rightarrow \infty} H_\ell(X[:k\ell] | Y[:k\ell])$$

### 4.2 Upper Bounding Dimension Using Entropy

For two sequences  $X \in \Sigma_1^\infty$  and  $Y \in \Sigma_2^\infty$ , let  $H(X|Y) = s$  for some  $s \in [0, 1]$ . We go on to prove that  $\dim_{FS}^Y(X) \leq s$ , by constructing a finite-state relative  $s'$  gale (for any  $s' > s$ ) that succeeds on  $X$  relative to  $Y$ . We design a gambler that places bets on strings of some length  $\ell$ , at which block entropy dips below the needed threshold. The bets are placed corresponding to the frequencies of appearance of these strings.

These frequencies need not always converge to a particular value, over the number of blocks considered. To overcome this, we pull out a subsequence for which the frequencies converge, simultaneously maintaining some desirable properties.

**Lemma 1.** For every  $X \in \Sigma_1^\infty$  and  $Y \in \Sigma_2^\infty$  with  $H(X|Y) = s$ ,  $\forall s' > s$ , there exists a block length  $\ell \in \mathbb{N}$ , and a function  $\mathbb{P}(x, y) : \Sigma_1^\ell \times \Sigma_2^\ell \rightarrow \mathbb{Q} \cap [0, 1]$  such that  $\sum_{x,y} \mathbb{P}(x, y) = 1$  and for infinitely many  $k \in \mathbb{N}$ ,

$$-\frac{1}{\ell \log(|\Sigma_1|)} \sum_{y \in \Sigma_2^\ell} P(y, Y[:k\ell]) \sum_{x \in \Sigma_1^\ell} P(x, X[:k\ell] | y, Y[:k\ell]) \log \mathbb{P}(x|y) < (s' - d)$$

for some  $0 < d < s'$ , where  $\mathbb{P}(x|y) = \frac{\mathbb{P}(x,y)}{\sum_x \mathbb{P}(x,y)}$ .

In Lemma 2, we construct a finite gambler  $G_\ell$  that effectively places the bet  $\mathbb{P}(x|y)$  on the string  $x$  when string provided by the oracle is  $y$ . It is shown that an  $s'$  gale corresponding to  $G_\ell$  succeeds in the required setting.

**Lemma 2.** For every  $X \in \Sigma_1^\infty$  and  $Y \in \Sigma_2^\infty$ ,  $\dim_{FS}^Y(X) \leq H(X|Y)$ .

### 4.3 Upper Bounding Entropy Using Dimension

For two sequences  $X \in \Sigma_1^\infty$  and  $Y \in \Sigma_2^\infty$ , let  $\dim_{FS}^Y(X) = s$  for some  $s \in [0, 1]$ . This happens when an  $s$ -gale  $d_{G,Y}^{(s)}$  corresponding to a finite-state gambler  $G$  succeeds on  $X$ , when given an oracle access to  $Y$ . For a block length  $L \in \mathbb{N}$ , we construct a stretched automaton  $G_L$ , to keep track of the state encountered after every run of input of length  $L$  on  $G$ . Using this, we show that for any sufficiently large block length  $L$ ,  $H_L(X|Y) \leq s + c/L$ , where  $c$  is a constant which does not depend on  $L$ . Therefore for any block length  $L$ , we get,

$$\limsup_{L \rightarrow \infty} H_L(X|Y) \leq \dim_{FS}^Y(X). \tag{1}$$

And hence we obtain the following lemma.

**Lemma 3.** *For every  $X \in \Sigma_1^\infty$  and  $Y \in \Sigma_2^\infty$ , we have  $H(X|Y) \leq \dim_{FS}^Y(X)$ .*

Lemma 2 established that  $\inf_L H_L(X|Y) \geq \dim_{FS}^Y(X)$  which implies that

$$\liminf_{L \rightarrow \infty} H_L(X|Y) \geq \dim_{FS}^Y(X). \tag{2}$$

**Theorem 1.**

$$\dim_{FS}^Y(X) = \lim_{\ell} \liminf_{k \rightarrow \infty} H_{\ell k}(X|Y).$$

*Proof.* This follows immediately from (1) and (2).

### 4.4 Multiple Oracles

Consider the case when the relative automaton has access to more than one oracle sequences, say  $Y_1 \cdots Y_n$ , where each  $Y_i \in \Sigma^\infty$ . The finite-state relative dimension in this scenario is defined using the case in which there is only one oracle  $Y \in (\Sigma^n)^\infty$ , such that  $Y = Y_1 \times Y_2 \cdots \times Y_n$ .

**Definition 6.** *(Finite-state relative dimension with multiple oracles) The finite-state relative dimension of a sequence  $X$  with respect to a sequences  $Y_1 \cdots Y_n$  is defined as  $\dim_{FS}^{Y_1 \cdots Y_n}(X) = \dim_{FS}^Y(X)$ .*

For a string  $X$  strings  $Y_1, \dots, Y_n \in \{0, 1\}^n$ , the  $\ell$ -length block entropy of  $X$  relative to  $Y_1, \dots, Y_n \in \{0, 1\}^n$  is defined as:

**Definition 7.** *(Relative block entropy rates with multiple oracles) The relative block entropy rate of  $X$  with respect to  $Y_1, \dots, Y_n$  is defined by  $H_\ell(X|Y_1 \cdots Y_n) = H_\ell(X|Y)$ .*

The proofs relating to Theorem 1 can also be stated in terms of the case where there are multiple oracle sequences involved.

**Theorem 2.**  $\dim_{FS}^{Y_1 \cdots Y_n}(X) = \lim_{\ell} \liminf_{k \rightarrow \infty} H_{\ell k}(X|Y_1 \cdots Y_n)$ .

The following are some basic properties of finite-state relative dimension.

**Lemma 4.** *Let  $X$  and  $Y$  be arbitrary strings in  $\Sigma^\infty$ . Then the following hold.*

1.  $\dim_{FS}^Y(X) \leq \dim_{FS}(X)$ .
2.  $\dim_{FS}^X(X) = 0$ .
3.  $\dim_{FS}^{0^\infty}(X) = \dim_{FS}(X)$ .

## 5 Finite-State Relative Dimensions of AP Subsequences

In this section, we study the finite-state dimensions of AP subsequences using the information theoretic characterization of finite-state relative dimension developed in Theorem 1. We derive a collection of inequalities which demonstrate the relationships between the dimension of a sequence and the dimensions of its AP subsequences. The main results in the following sections are derived as the consequences of these inequalities. At the end of this section, we show that these bounds are necessarily inequalities, by providing examples. These examples also show that the bounds are tight.

Given a sequence  $X \in \Sigma^\infty$ ,  $d \in \mathbb{N}$  and any  $i \in \{0, 1, 2, \dots, d - 1\}$  let  $A_{d,i}$  denote the  $i^{\text{th}}$  AP subsequence of  $X$  with common difference  $d$ . That is,

$$A_{d,i} = X[i] X[i + d] X[i + 2d] X[i + 3d] \dots$$

Using the chain rule of Shannon entropy [6], we obtain our first inequality which gives a lower bound for the finite-state dimension of  $X$  in terms of the finite-state relative dimensions of the its AP subsequences.

**Lemma 5.** *For any  $X \in \Sigma^\infty$ ,  $d \in \mathbb{N}$  and  $\sigma \in \text{Bij}(d)$ ,*

$$\dim_{FS}(X) \geq \frac{1}{d} \left( \dim_{FS}(A_{d,\sigma(0)}) + \sum_{i=1}^{d-1} \dim_{FS}^{A_{d,\sigma(0)}, A_{d,\sigma(1)}, \dots, A_{d,\sigma(i-1)}}(A_{d,\sigma(i)}) \right)$$

Using inequalities from real analysis, and by selecting  $\sigma$  to be the identity mapping, we can show the following corollary.

**Corollary 1.** *Consider any  $X \in \Sigma^\infty$  and  $d \in \mathbb{N}$ . If for some  $r \in [0, 1]$ ,  $\dim_{FS}(A_{d,0}) \geq r$  and for every  $i$ ,*

$$\dim_{FS}^{A_{d,0}, A_{d,1}, \dots, A_{d,i-1}}(A_{d,i}) \geq r.$$

*Then,  $\dim_{FS}(X) \geq r$ .*

It follows from Theorem 4 in Sect. 6 that when  $X$  is a normal sequence, the bound in Lemma 5 is tight. To conclude this section, we show that inequality in Lemma 5 cannot be replaced with an equality in general. We state the following lemma for the case when  $d = 2$  and  $\sigma$  is the identity mapping from  $\{0, 1\}$  to itself, for the sake of simplicity. Using ideas from [11], we obtain the following lemma which generalizes to arbitrary  $d$  and  $\sigma$  in a straightforward manner.

**Lemma 6.** *There exists an  $X \in \Sigma^\infty$  such that,*

$$\dim_{FS}(X) > \frac{1}{2} \left( \dim_{FS}(A_{2,0}) + \dim_{FS}^{A_{2,0}}(A_{2,1}) \right)$$



## 6 A Stronger Wall's Theorem on AP Subsequences with a Perfect Converse

D.D.Wall in [14] proved that a number  $x = 0.x[0]x[1]x[2]x[3]\dots$  is normal if and only if for every  $d \geq 1$  and  $a \geq 0$ ,  $0.x[a]x[a+2d]x[a+3d]x[a+4d]\dots$  is a normal number. We state the equivalent theorem in  $\Sigma^\infty$  below.

**Theorem 3.** [14] *If  $X \in \Sigma^\infty$  is a normal sequence then for every  $d \geq 1$  and  $a \geq 0$ ,  $X[a] X[a+d] X[a+2d] X[a+3d] X[a+4d]\dots$  is a normal sequence.*

We remark that it is enough to consider  $a \in \{0, 1, 2, \dots, d-1\}$  since replacing  $a$  with  $a \bmod d$  only prepends finitely many characters to the subsequence (and therefore has no effect on the normality or the finite-state dimension of the subsequence). If  $d = 1$ , then the converse direction is trivial since the AP subsequence with  $a = 0$  and  $d = 1$  is the sequence  $X$  itself. Therefore, the converse direction is interesting only when  $d$  takes values strictly greater than 1. Hence, the new converse question is the following: *If for every  $d \geq 2$  and  $a \in \{0, 1, 2, \dots, d-1\}$ ,  $X[a] X[a+d] X[a+2d] X[a+3d] X[a+4d]\dots$  is a normal sequence then is  $X$  a normal sequence?*

This question is shown to be false. Vandehey [13] gave the following counterexample: fix any normal sequence  $X = X[0]X[1]X[2]X[3]\dots$  and consider the *doubled sequence*,  $X[0]X[0]X[1]X[1]X[2]X[2]X[3]X[3]\dots$ . It is straightforward to verify that every AP subsequence of the *doubled sequence* with  $d \geq 2$  is normal, but the *doubled sequence* is itself non-normal.

Weiss [15], Kamae [9] and Vandehey [13] showed that expanding the set of subsequences along which normality is investigated can yield interesting answers in the converse direction. Weiss and Kamae [9, 15] showed that a number is normal if and only normality is preserved along every *deterministic* subsequence with positive asymptotic lower density. In a recent work, Vandehey [13] proved a nearly sharp converse to the Wall's theorem. Vandehey considers collections of subsequences such that for any  $\epsilon$ , the collection contains a subsequence with asymptotic lower density greater than  $1 - \epsilon$ . Theorem 1.3 in [13] shows that preservation of normality along all subsequences in such a collection implies the normality of the original number. Theorem 1.4 from the same paper shows that this converse to the Wall's theorem is close to being sharp.

In this section we show that the inequalities established in Sect. 5 yields a *stronger* forward direction for Wall's theorem (Theorem 3). We show that if sequence  $X$  is normal, then for any  $d \geq 2$  and  $a \in \{0, 1, 2, \dots, d-1\}$ , the AP subsequences  $X[a] X[a+d] X[a+2d] X[a+3d] X[a+4d]\dots$  is *relatively normal* with respect to every other AP subsequence with the same common difference  $d$ . The results in Sect. 5 also yields a perfect converse to the *stronger* forward direction. i.e., we show that if every AP subsequence of  $X$  with  $d \geq 2$  is *relatively normal* with respect to every other AP subsequence with the same common difference  $d$ , then  $X$  is a normal sequence.

For proving the forward direction we require the following lemma.

**Lemma 7.** For any  $X \in \Sigma^\infty$ ,  $d \in \mathbb{N}$  and  $j \in \{0, 1, 2, \dots, d - 1\}$ ,

$$\dim_{FS}(A_{d,j}) \geq \dim_{FS}^{\{A_{d,k} | k \neq j\}}(A_{d,j}) \geq d \left( \dim_{FS}(X) - \frac{d-1}{d} \right).$$

Now, we give an immediate corollary of Lemma 7 which is useful in the later sections.

**Corollary 2.** Let  $X \in \Sigma^\infty$ ,  $d \in \mathbb{N}$  and  $j \in \{0, 1, 2, \dots, d - 1\}$ . If for some  $\epsilon > 0$ ,

$$\dim_{FS}(X) \geq \frac{d-1}{d} + \epsilon,$$

then for every  $j$ ,  $\dim_{FS}(A_{d,j}) \geq \dim_{FS}^{\{A_{d,k} | k \neq j\}}(A_{d,j}) \geq d\epsilon$ .

The following is the restatement of Theorem 3 in terms of finite-state dimension which easily follows from the entropy characterization of finite-state dimension from [3].

**Theorem (Restatement of Theorem 3).** A sequence  $X \in \Sigma^\infty$  is such that  $\dim_{FS}(X) = 1$  if and only if for every  $a \in \{0, 1, 2, \dots, d - 1\}$  and  $d \geq 2$ ,  $\dim_{FS}(A_{d,a}) = 1$ .

As a consequence of Corollary 2, by setting  $\epsilon = 1/d$ , we obtain the stronger forward direction of Theorem 3.

**Lemma 8.** Let  $X \in \Sigma^\infty$  be any normal sequence. Then, for every  $d \geq 2$  and  $a \in \{0, 1, 2, \dots, d - 1\}$ ,  $\dim_{FS}(A_{d,a}) = \dim_{FS}^{\{A_{d,k} | k \neq a\}}(A_{d,a}) = 1$ .

The above statement strengthens the forward direction of Theorem 3 because it claims that if a sequence is normal then each of its AP subsequences are normal and are also relatively normal with respect to the other AP subsequences having the same common difference.

The conclusion in Lemma 8 is strong enough that using our lower bound inequality (Lemma 5), we get a perfect converse to Lemma 8.

**Lemma 9.** Let  $X \in \Sigma^\infty$ . If for every  $d \geq 2$  and  $a \in \{0, 1, 2, \dots, d - 1\}$ ,  $\dim_{FS}^{\{A_{d,k} | k \neq a\}}(A_{d,a}) = 1$ , then  $\dim_{FS}(X) = 1$ .

Lemma 9 follows using the fact that  $\dim_{FS}^{A_{d,0}, A_{d,1}, \dots, A_{d,a-1}}(A_{d,a}) \geq \dim_{FS}^{\{A_{d,k} | k \neq a\}}(A_{d,a})$  and Corollary 1 when  $r = 1$ .

Combining Lemma 8 and 9, we get the following stronger Wall’s theorem with a perfect converse.

**Theorem 4.** A sequence  $X \in \Sigma^\infty$  is normal (equivalently  $\dim_{FS}(X) = 1$ ) if and only if for every  $d \geq 2$  and  $a \in \{0, 1, 2, \dots, d - 1\}$ ,  $\dim_{FS}^{\{A_{d,k} | k \neq a\}}(A_{d,a}) = 1$ .

In other words, we have shown that a sequence  $X \in \Sigma^\infty$  is normal if and only if for every  $d \geq 2$ , the AP subsequences of  $X$  with common difference  $d$  are normal and are also relatively normal with respect to the other AP subsequences with the same common difference  $d$ .

## 7 van Lambalgen’s Theorem for Finite-State Dimension

van Lambalgen, in his thesis, [10] showed that relative Martin-Löf randomness is symmetric.

**Theorem 5.** [10] *Let  $A, B \in \Sigma^\infty$ .  $A$  is Martin-Löf random and  $B$  is Martin-Löf random relative to  $A$  if and only if  $A \oplus B$  is Martin-Löf random.*

This symmetry fails in other randomness settings like Schnorr randomness, computable randomness and resource-bounded randomness [1, 5, 8, 16]. In this section, we show that relative normality is symmetric, thus establishing an analogue of van Lambalgen’s theorem for normality. But, for finite-state dimensions less than 1, both directions of the theorem fails to hold in general. We show that for the class of regular sequences, the forward direction of van Lambalgen’s Theorem is true.

Utilizing the results we established in Sects. 5 and 6, we first show that an analogue of van Lambalgen’s theorem holds for normality, *i.e.* the case when finite-state dimensions are 1.

**Theorem 6.** *Let  $A, B \in \Sigma^\infty$ .  $A$  is normal and  $B$  is normal relative to  $A$  if and only if  $A \oplus B$  is normal.*

*Proof.*  $A$  and  $B$  are two A.P. subsequences of  $A \oplus B$  with common difference 1. By Corollary 1 it follows that  $\dim_{FS}(A \oplus B) = 1$ , *i.e.*,  $A \oplus B$  is normal.

Conversely, suppose that  $A \oplus B$  is normal, *i.e.*  $\dim_{FS}(A \oplus B) = 1$ . By Lemma 8,  $\dim_{FS}^A(B) = 1$  and  $\dim_{FS}^B(A) = 1$ . Since  $\dim_{FS}(A) \geq \dim_{FS}^B(A)$  by Lemma 7, it follows that  $\dim_{FS}(A) = 1$ .

Thus relative normality is symmetric. We may conjecture that this generalizes in the following ideal form.

**Ideal Claim.** Let  $A, B \in \Sigma^\infty$ . Then for any  $r \in [0, 1]$ ,  $\dim_{FS}(A) = r$  and  $\dim_{FS}^A(B) = r$  if and only if  $\dim_{FS}(A \oplus B) = r$ .

Both the forward and converse directions in the above claim are false for general sequences. However, we conclude by showing that if  $A$  and  $B$  are “regular sequences”, then the forward direction of the ideal claim holds. This is analogous to the failure of the converse direction of van Lambalgen’s theorem in certain notions of randomness.

Setting  $A$  and  $B$  to be the two AP subsequences (with  $d = 2$ ) of  $X$  from Lemma 6, it readily follows that the forward direction in the above claim is false for general  $A$  and  $B$ . Now let  $A$  be  $0^\infty$  and  $B$  be any normal number. Now,  $A \oplus B$  is the diluted sequence [7] with dimension equal to  $1/2$ . But,  $\dim_{FS}(A) = 0$  and it follows from Lemma 4 that  $\dim_{FS}^A(B) = 1$ . Therefore the converse direction of the claim is also false for general sequences. It is easy to verify from the construction in the proof of Lemma 6 that sequences  $A$  and  $B$  given in the counterexample for the forward direction are both non-regular sequences. This leads to the question whether the forward direction in the ideal claim is true for

$A$  and  $B$  that are regular sequences. We answer this question in the affirmative as a consequence of the following lemma.

**Lemma 10.** *Let  $A, B \in \Sigma^\infty$ . If  $A$  is a regular sequence, then,*

$$\dim_{FS}(A \oplus B) = \frac{1}{2} \left( \dim_{FS}(A) + \dim_{FS}^A(B) \right).$$

Now we prove the forward direction of the ideal claim for regular sequences.

**Lemma 11.** *Let  $A, B \in \Sigma^\infty$  such that  $A$  is a regular sequence and let  $r \in [0, 1]$ . If  $\dim_{FS}(A) = r$  and  $\dim_{FS}^A(B) = r$  then,  $\dim_{FS}(A \oplus B) = r$ .*

The converse of the ideal claim is however false even for  $A$  and  $B$  that are both regular sequences. Let  $A$  be  $0^\infty$  and  $B$  be any normal number. Both of these sequences are regular sequences. However, as we noted above,  $\dim_{FS}(A \oplus B) = 1/2$ , but  $\dim_{FS}(A) = 0$  and  $\dim_{FS}^A(B) = 1$ .

## 8 Relation to Finite-State Independence

Becher, Carton and Heiber [2] define finite-state independence between two strings. They use a joint compression model composing of a one-to-one finite-state transducer with auxiliary input. Two strings  $X$  and  $Y$  are said to be finite-state independent when one does not help to compress the other in this model. However, due to the presence of  $\varepsilon$ -transitions, the input tape of  $X$  and  $Y$  need not be read in tandem. Our model is more restrictive, hence more pairs of sequences are relatively finite-state random in our model. This is a consequence of the theorems in the previous section.

**Lemma 12.** *There are sequences  $X$  and  $Y$  that are not finite-state independent but  $X$  and  $Y$  are relatively normal, and  $X \oplus Y$  is normal.*

*Proof.* By Theorem 4.3 in [2], there exists two normal strings  $X$  and  $Y$  such that  $X \oplus Y$  is normal but  $X$  and  $Y$  are not finite-state independent. Since  $\dim_{FS}(X \oplus Y) = 1$ , by Theorem 6, we have  $\dim^X(Y) = 1$ .

The converse question, *i.e.* whether finite-state independence implies relative randomness, remains open.

## References

1. Bauwens, B.: Uniform van Lambalgen's theorem fails for computable randomness. *Inf. Comput.* **271**, 104486 (2020). <https://doi.org/10.1016/j.ic.2019.104486>
2. Becher, V., Carton, O., Heiber, P.A.: Finite-state independence. *Theor. Comput. Syst.* **62**(7), 1555–1572 (2017). <https://doi.org/10.1007/s00224-017-9821-6>
3. Bourke, C., Hitchcock, J.M., Vinodchandran, N.: Entropy rates and finite-state dimension. *Theor. Comput. Sci.* **349**(3), 392–406 (2005)
4. Case, A., Lutz, J.H.: Finite-state mutual dimension. *arXiv preprint arXiv:2109.14574* (2021)

5. Chakraborty, D., Nandakumar, S., Shukla, H.: On resource-bounded versions of the van Lambalgen Theorem. In: Gopal, T.V., Jäger, G., Steila, S. (eds.) TAMC 2017. LNCS, vol. 10185, pp. 129–143. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-55911-7\\_10](https://doi.org/10.1007/978-3-319-55911-7_10)
6. Cover, T.M., Thomas, J.A.: Elements of information theory. Wiley Series in Telecommunications. Wiley, New York (1991). <https://doi.org/10.1002/0471200611>. a Wiley-Interscience Publication
7. Dai, J.J., Lathrop, J.I., Lutz, J.H., Mayordomo, E.: Finite-state dimension. In: Orejas, F., Spirakis, P.G., van Leeuwen, J. (eds.) ICALP 2001. LNCS, vol. 2076, pp. 1028–1039. Springer, Heidelberg (2001). [https://doi.org/10.1007/3-540-48224-5\\_83](https://doi.org/10.1007/3-540-48224-5_83)
8. Downey, R.G., Hirschfeldt, D.R.: Algorithmic randomness and complexity. Theory and Applications of Computability, Springer, New York (2010). <https://doi.org/10.1007/978-0-387-68441-3>
9. Kamae, T.: Subsequences of normal sequences. Israel J. Math. **16**, 121–149 (1973). <https://doi.org/10.1007/BF02757864>
10. Lambalgen, M.V.: Random sequences. Amsterdam: Academish Proefshrit (1987)
11. Lutz, J.H., Nandakumar, S., Pulari, S.: A weyl criterion for finite-state dimension. arXiv preprint [arXiv:2111.04030](https://arxiv.org/abs/2111.04030) (2021)
12. Schnorr, C.P., Stimm, H.: Endliche automaten und zufallsfolgen. Acta Informatica **1**, 345–359 (1972)
13. Vandehey, J.: Towards a sharp converse of Wall’s theorem on arithmetic progressions. Pacific J. Math. **300**(2), 499–509 (2019). <https://doi.org/10.2140/pjm.2019.300.499>
14. Wall, D.D.: NORMAL NUMBERS. ProQuest LLC, Ann Arbor, MI (1950), thesis (Ph.D.)-University of California, Berkeley
15. Weiss, B.: Normal sequences as collectives. In: Proceedings of Symposium on Topological Dynamics and Ergodic Theory, University of Kentucky (1971)
16. Yu, L.: When van Lambalgen’s theorem fails. Proc. Amer. Math. Soc. **135**(3), 861–864 (2007). <https://doi.org/10.1090/S0002-9939-06-08541-8>



# Distributed Connected Dominating Sets in Unit Square and Disk Graphs

Barun Gorain<sup>1</sup>, Kaushik Mondal<sup>2</sup>, and Supantha Pandit<sup>3</sup>(✉)

<sup>1</sup> Indian Institute of Technology Bhilai, Raipur, Chattisgarh, India  
barun@iitbhillai.ac.in

<sup>2</sup> Indian Institute of Technology Ropar, Rupnagar, Punjab, India  
kaushik.mondal@iitrpr.ac.in

<sup>3</sup> Dhirubhai Ambani Institute of Information and Communication Technology,  
Gandhinagar, Gujarat, India  
pantha.pandit@gmail.com

**Abstract.** The Minimum Dominating Set (MDS) and Minimum Connected Dominating set (MCDS) problems are well-studied problems in the distributed computing communities due to their numerous applications across the field. We study these problems in axis-parallel unit square and unit disk graphs. We exploit the underlying geometric structures of these graph classes and present constant round distributed algorithms in the *LOCAL* communication model. Our results are distributed constant factor approximation algorithms for the MCDS problem in unit square graphs that run in 18 rounds and in unit disk graphs that run in 44 rounds. The message complexity is linear for both the algorithms.

**Keywords:** Minimum dominating set · Minimum connected dominating set · Distributed algorithms · Axis-parallel unit square graphs · Unit disk graphs · Approximation algorithms

## 1 Introduction

The dominating set problem is a well-studied problem in combinatorial optimization. This problem has numerous real-life applications in facility location, wireless networking problems, and many more. For an undirected graph  $G = (V, E)$ , a subset  $S \subseteq V$  is called a dominating set of  $G$ , if for any vertex  $v \in V$ , either  $v \in S$  or there exists a node  $u \in S$  and  $(u, v) \in E$ . The minimum dominating set (MDS) problem seeks to find a dominating set in  $G$  of minimum size. A connected dominating set in  $G$  is a dominating set whose induced graph is connected. The

---

The authors acknowledge the support of the Science and Engineering Research Board (SERB), Department of Science and Technology, Govt. of India (grant no. CRG/2020/005964). Dr. Barun Gorain also acknowledges the support of the Science and Engineering Research Board (SERB), Department of Science and Technology, Govt. of India (grant no. MTR/2021/000118), and the Research Initiation Grant supported by the Indian Institute of Technology Bhilai, India.

minimum connected dominating set (MCDS) problem seeks a connected dominating set of minimum size. Both the MDS and the MCDS problems are known to be NP-complete.

In a geometric setting, we are given a set  $\mathcal{O}$  of objects in the Euclidean plane. The objective of the MDS problem is to find a minimum size subset  $\mathcal{O}' \subseteq \mathcal{O}$  such that for any object  $o \in \mathcal{O}$ , either  $o \in \mathcal{O}'$  or there exists at least one object  $o' \in \mathcal{O}'$  such that  $o$  and  $o'$  intersect. Similarly, we define the MCDS problem where the objects in  $\mathcal{O}'$  form a connected component. We now define the geometric intersection graph  $G_{\mathcal{O}}$  of the set  $\mathcal{O}$ . For each object  $o \in \mathcal{O}$ , take a vertex  $v_o \in G_{\mathcal{O}}$ . There is an edge in  $G_{\mathcal{O}}$  between two vertices  $v_{o_1}$  and  $v_{o_2}$  if and only if the two objects  $o_1, o_2 \in \mathcal{O}$  intersect. For our convenience, we use nodes and objects interchangeably when it is clear from the context to represent an object and its corresponding node in the intersection graph of the objects. In all algorithms in this paper, we assume that each vertex knows its position.

In many real-life applications, the underlying graphs have geometric structure. For example, in a wireless sensor network, sensor nodes are deployed over an unknown region for data collection. Every sensor node has a communication range  $r_c$  and a node can communicate with all the nodes which are inside its communication range. Thus, every sensor node can be represented by a disk of radius  $r_c/2$ . For any two sensor nodes which are less than  $r_c$  distance to each other, the corresponding disks intersect and the nodes can communicate to each other. Hence, the underlying communication graph can be represented by a geometric graph, where each vertex is a disk and there is an edge between two vertices if the corresponding disks intersect.

In a distributed setting, the MCDS problem is well-studied. It is used as backbone in ad-hoc and sensor networks [6, 7]. Connected dominating sets are used for routing, broadcasting and other network management functions in a network [6]. In this paper we provide distributed algorithms for the MDS and MCDS problems on axis-parallel unit squares graphs, and unit disk graphs. Our algorithms run in constant rounds. Further we show that these algorithms produce constant factor approximations using no more than linear number of messages.

We consider the well-known *LOCAL* communication model [16]. Communication proceeds in synchronous rounds and all nodes start simultaneously.<sup>1</sup> In any round, a node can send an arbitrary size message to all of its neighbors and can perform any local computations. The number of such synchronous rounds required till an algorithm ends is the time complexity of the algorithm. The same model is considered in the work by Jallu et.al. [11]. Regarding communication, when a node broadcasts, we consider it as a single message sent by the node, similar to the model considered in [11]. The total number of messages sent by all the nodes during the algorithm is the message complexity of the algorithm.

## 1.1 Previous Work

The MDS problem is NP-hard for simple geometric objects such as axis-parallel unit squares and unit disks [5]. The MCDS problem is NP-hard for unit disk

<sup>1</sup> In an asynchronous network, simulating *LOCAL* model using time stamps is possible.

graphs [12]. In the sequential setting, constant factor approximation algorithms for the MDS and MCDS problems are given in [2, 13]. For unit disk graphs, PTASes exist for the MDS [10] and MCDS problems [4]. It is straightforward that simple greedy algorithms can be designed for the MDS problem for unit square and disk graphs.

There is a series of study of the connected dominating set problem in the distributed settings [3, 8, 14, 17–19]. There are some algorithms for the MCDS problem in unit disk graphs that give low constant approximation algorithms (factor 8), however, their time complexity is linear and message complexity is  $O(n \log n)$ , where  $n$  is the number of nodes in the graph [3, 17]. Alzoubi et al. [1], Cheng et al. [3], and Gao et al. [9] proposed constant factor approximation algorithms for the MCDS problem in unit disk graphs and both time and message complexity of the algorithms are linear. Jallu et al. [11] provided an algorithm with  $(104opt + 52)$  approximation for the MCDS problem in unit disk graphs that takes  $O(\Delta)$  time and  $O(n)$  messages, where  $\Delta$  is the maximum degree of a node in the graph and  $n$  is the number of nodes. Authors also dealt with the interference issue while broadcasting by scheduling conflict-free time slots for the nodes to broadcast. Mohanty et al. [15] provided  $((4.8 + \log 5)opt + 52)$  approximation for the MCDS problem in unit disk graphs in  $O(D)$  rounds which is very high considering  $D$  is the diameter of the graph.

## 1.2 Our Contributions

- Axis-parallel unit square graphs
  - We present a 18 rounds distributed approximation algorithm for the MCDS problem in axis-parallel unit square graphs. Next, we show that this algorithm is a 72 factor approximation algorithm for the MCDS problem in axis-parallel unit square graphs. The message complexity is linear in the number of nodes. (Sect. 2)
- Unit disk graphs
  - We present a 44 rounds distributed approximation algorithm for the MCDS problem in the unit disk graphs. We further show that this algorithm is a 441 factor approximation algorithm for the MCDS problem in unit disk graph. The message complexity is linear. (Sect. 3)

## 2 Unit Square Graphs

In this section, we present a constant round distributed algorithm for finding a connected dominating set in an axis-parallel unit square intersection graph. Let  $S$  be a given set of axis-parallel unit squares. Each square in  $S$  is identified by the coordinates of its lower-left corner which we refer to as its *id*. Consider the vertical lines (i.e., lines parallel to the  $y$ -axis) at each integer point on the  $x$ -axis. Similarly, consider the horizontal lines (i.e., lines parallel to the  $x$ -axis) at each integer point on the  $y$ -axis. These lines form a unit grid  $R$  on the  $xy$ -plane (refer to Fig. 1). Each grid point can be identified by its coordinate  $(i, j)$  where vertical line at integer  $i$  and horizontal line at integer  $j$  intersects.



A point  $(x, y)$  is said to be covered by a square  $s \in S$  with lower left corner  $(p, q)$  if  $p \leq x < p+1$  and  $q \leq y < q+1$ . As per the above definition, each square  $s \in S$  covers exactly one grid point. Let  $S_{ij} \in S$  be the set of squares that cover the grid point  $(i, j)$ . Let  $C_L$  be the resulting minimum connected dominating set returned by the algorithm that is empty at the beginning of the algorithm. The intuitive idea behind the algorithm is demonstrated in Fig. 1.

### 2.1 The Algorithm

The proposed algorithm runs for 18 rounds that proceed in three phases. The first phase involves the initial two rounds. A set of nodes in  $S_{ij}$  selects themselves in the dominating set in this phase. The next two phases, each of which takes 8 rounds are designated to select few more nodes to ensure the connectivity.

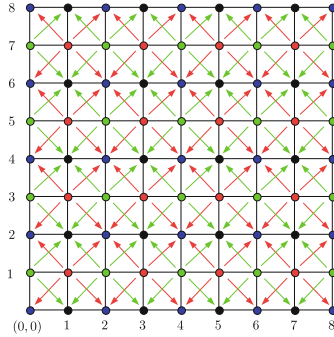
We introduce some notations use in the algorithm. For any node  $v$ ,  $N(v)$  is the set of neighbors of  $v$ , including  $v$ . Let  $N'(v) \subseteq N(v)$  be the set of neighbors of  $v$ , including  $v$ , that contains the same integer grid point as  $v$ . For any non-empty subset  $T$  of  $S$ ,  $y_{min}(T)$  is the node  $w$  in  $T$  such that the  $y$ -coordinate of  $w$  is minimum among all the nodes in  $T$ . In case of tie, the node  $w$  is chosen as the node with minimum  $x$  coordinate among all the nodes which are involved in the tie. For any non-empty subset  $T$  of  $V$ ,  $y_{max}(T)$  is the node  $w$  in  $T$  such that the  $y$ -coordinate of  $w$  is maximum among all the nodes in  $T$ . In case of tie, the node  $w$  is chosen as the node with maximum  $x$  coordinate among all the nodes which are involved in the tie. Similarly, we define  $x_{min}(T)$  and  $x_{max}(T)$ .

**Phase 1:** Each node  $v$  broadcasts its  $id$  and thus learns  $N(v)$  in the first round. In round 2, the node  $v$  computes the functions  $x_{min}(N'(v))$ ,  $y_{min}(N'(v))$ ,  $x_{max}(N'(v))$ ,  $y_{max}(N'(v))$  after learning  $N(v)$  and computing  $N'(v)$  from  $N(v)$ . The node  $v$  selects itself in  $C_L$  if  $v$  itself is one of the nodes  $x_{min}(N'(v))$ ,  $y_{min}(N'(v))$ ,  $x_{max}(N'(v))$ ,  $y_{max}(N'(v))$ .

**Phase 2:** All the nodes in  $S_{ij}$ , where both  $i, j$  are odd, participate in this phase.

- Each node  $v$  in  $S_{ij}$  that has at least one neighbor in  $S_{(i+1)(j+1)}$  broadcasts its  $id$  in round 1 of this phase. In round 2, the node  $v$  learns the set  $N(i+1, j+1)(v) \subseteq S_{ij}$  containing the set of all neighbors which has at least one neighbor in  $S_{(i+1)(j+1)}$ . If  $v \in N(i+1, j+1)(v)$  and  $x_{min}(N(i+1, j+1)(v)) = id(v)$ , then  $v$  selects itself in  $C_L$  and broadcasts its  $id$ .
- Each node  $v$  in  $S_{ij}$  that has at least one neighbor in  $S_{(i+1)(j-1)}$  broadcasts its  $id$  in round 3 of this phase. In round 4, the node  $v$  learns the set  $N(i+1, j-1)(v) \subseteq S_{ij}$  containing the set of all neighbors which has at least one neighbor in  $S_{(i+1)(j-1)}$ . If  $v \in N(i+1, j-1)(v)$  and  $x_{min}(N(i+1, j-1)(v)) = id(v)$ , then  $v$  select itself in  $C_L$  and broadcasts its  $id$ .
- Each node  $v$  in  $S_{ij}$  that has at least one neighbor in  $S_{(i-1)(j-1)}$  broadcasts its  $id$  in round 5 of this phase. In round 6, the node  $v$  learns the set  $N(i-1, j-1)(v) \subseteq S_{ij}$  containing the set of all neighbors which has at least one neighbor in  $S_{(i-1)(j-1)}$ . If  $v \in N(i-1, j-1)(v)$  and  $x_{min}(N(i-1, j-1)(v)) = id(v)$ , then  $v$  select itself in  $C_L$  and broadcasts its  $id$ .

- Each node  $v$  in  $S_{ij}$  that has at least one neighbor in  $S_{(i-1)(j+1)}$  broadcasts its  $id$  in round 7 of this phase. In round 8, the node  $v$  learns the set  $N(i-1, j+1)(v) \subseteq S_{ij}$  containing the set of all neighbors which has at least one neighbor in  $S_{(i-1)(j+1)}$ . If  $v \in N(i-1, j+1)(v)$  and  $x_{min}(N(i-1, j+1)(v)) = id(v)$ , then  $v$  select itself in  $C_L$  and broadcasts its  $id$ .



**Fig. 1.** Grid points  $\{(i, j)|i, j \text{ odd}\}$ ,  $\{(i, j)|i \text{ odd}, j \text{ even}\}$ ,  $\{(i, j)|i \text{ even}, j \text{ odd}\}$ , and  $\{(i, j)|i, j \text{ even}\}$  are colored with red, green, black, and blue, respectively. In phase 1, the selected node corresponding to any grid point (e.g., red) becomes connected to the selected nodes corresponding to its two horizontal and vertical neighboring grid points (e.g., two green and two black) if there is a direct connection. In phase 2, nodes corresponding to red grid points make connection with nodes corresponding to blue grid points that is interpreted by the red arrows. In Phase 3, nodes corresponding to green grid points make connection with nodes corresponding to black grid points, that is represented by the green arrows. (Color figure online)

**Phase 3:** In this phase only the nodes in  $S_{ij}$  for  $i$  even and  $j$  odd, participate.

- Each node  $v$  in  $S_{ij}$  which received a message in the second round of Phase 2 from a node in  $S_{(i-1)(j-1)}$  broadcasts its  $id$  in round 1 of this phase. In round 2, the node  $v$  learns the set  $S'' \subseteq S_{ij}$  containing the set of all neighbors which has received a message in the 2nd round of phase 2 from a node in  $S_{(i-1)(j-1)}$ . If  $v \in S''$  and  $x_{min}(S'') = id(v)$ , then  $v$  select itself in  $C_L$ .
- Each node  $v$  in  $S_{ij}$  which received a message in the fourth round of Phase 2 from a node in  $S_{(i-1)(j+1)}$  broadcasts its  $id$  in round 3 of this phase. In round 4, the node  $v$  learns the set  $S'' \subseteq S_{ij}$  containing the set of all neighbors which has received a message in the fourth round of phase 2 from a node in  $S_{(i-1)(j+1)}$ . If  $v \in S''$  and  $x_{min}(S'') = id(v)$ , then  $v$  select itself in  $C_L$ .
- Each node  $v$  in  $S_{ij}$  which received a message in the sixth round of Phase 2 from a node in  $S_{(i+1)(j+1)}$  broadcasts its  $id$  in round 5 of this phase. In round 6, the node  $v$  learns the set  $S'' \subseteq S_{ij}$  containing the set of all neighbors which has received a message in the sixth round of phase 2 from a node in  $S_{(i+1)(j+1)}$ . If  $v \in S''$  and  $x_{min}(S'') = id(v)$ , then  $v$  select itself in  $C_L$ .

- Each node  $v$  in  $S_{ij}$  which received a message in the eighth round of Phase 2 from a node in  $S_{(i+1)(j-1)}$  broadcasts its  $id$  in round 7 of this phase. In round 8, the node  $v$  learns the set  $S'' \subseteq S_{ij}$  containing the set of all neighbors which has received a message in the eighth round of phase 2 from a node in  $S_{(i+1)(j-1)}$ . If  $v \in S''$  and  $x_{min}(S'') = id(v)$ , then  $v$  select itself in  $C_L$ .

A pictorial description of the algorithm is given in Fig. 1. The following theorems give the correctness and the approximation factor of the algorithm.

**Theorem 1.** *The nodes in  $C_L$  form a connected dominating set.*

**Theorem 2.** *The proposed algorithm returns a 72-approximation connected dominating set in 18 rounds using  $O(n)$  messages where  $n$  is the number of nodes.*

### 3 Unit Disk Graphs

In this section, we propose a distributed constant rounds algorithm to find connected dominating set for connected networks modeled as unit disk graphs (udg) where each unit disk is of radius 1 and identified by its center where the center is interpreted as the node corresponding to that unit disk. In these networks, a node  $u$  is connected to all the nodes at distance at most 1 from it.

#### 3.1 High Level Idea

Before going to the details of the algorithm for finding a minimal connected dominating set of  $G$ , we give a high level idea of the algorithm.

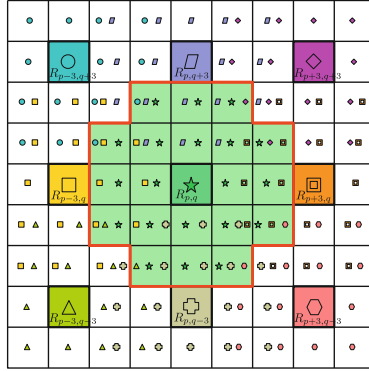
First, we divide the plane into squares of side lengths  $\frac{1}{\sqrt{2}}$ . This can be done by considering vertical and horizontal lines with  $\frac{1}{\sqrt{2}}$  distance apart over the entire region. Among the horizontal lines, one is along the  $x$ -axis and among the vertical lines, one is along the  $y$ -axis. For any two integers  $p, q$ , the square whose bottom left corner point is  $(p/\sqrt{2}, q/\sqrt{2})$ , is called  $(p, q)$ -th square and is denoted by  $R_{p,q}$ . A node  $u$  is said to be in the square  $R_{p,q}$  if it is inside the square or on the left or bottom boundary including the bottom left corner point. It can be noted here that by doing local computation, each node can learn in which square it belongs as the node knows its coordinates.

Intuitively, if one node is selected from each of the squares, then the set of selected nodes form a dominating set. This is because the side-length of any square is  $\frac{1}{\sqrt{2}}$  and hence, the length of its diagonal is 1. Therefore, any two nodes from a single square must be neighbors. As our objective is to find a connected dominating set, some additional nodes need to be selected. We select these additional nodes locally by identifying some squares as *head squares* and a set of squares around each head square as the *family* of that head square. The head squares are defined in a recursive way as follows. Refer to Fig. 2.

- The square  $R_{1,1}$  is a head square.
- The squares  $R_{p,q+3}$ ,  $R_{p+3,q}$ ,  $R_{p-3,q}$ , and  $R_{p,q-3}$  are head squares if and only if  $R_{p,q}$  is a head square.

For any square  $R_{i,j}$ , we define the family  $\mathfrak{F}(R_{i,j})$  of  $R_{i,j}$  as the set of squares  $\{R_{x,y} : (|x - i| \leq 2 \text{ and } |y - j| \leq 1) \text{ or } (|x - i| \leq 1 \text{ and } |y - j| \leq 2)\}$ . Note that  $|\mathfrak{F}(R_{i,j})| = 21$  (See Fig. 3a). Observe that, any square  $R_{i,j}$  can be part of at most three families of different head squares (refer Fig. 2). Let  $C(R_{i,j}) = \{(p, q) : R_{i,j} \in \mathfrak{F}(R_{p,q}), \text{ where } R_{p,q} \text{ is a head square}\}$ .

It can be observed that for any node  $u \in R_{i,j}$ , each node in  $N(u)$  belongs to some square in  $\mathfrak{F}(R_{i,j})$ . This is because the distance between  $u$  and any point outside  $\mathfrak{F}(R_{i,j})$  is more than 1 (See Fig. 3b).



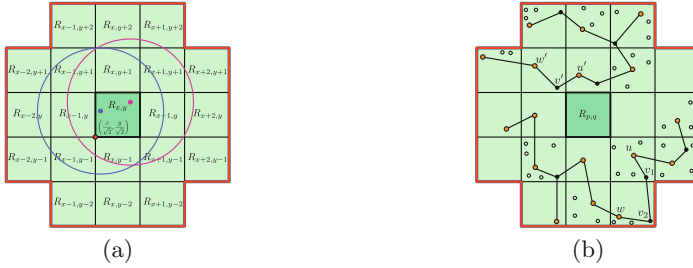
**Fig. 2.** We show a part of the Euclidean plane divided with vertical and horizontal lines. Each of the 9 head squares present in this part are shown using different colors along with a mark inside; e.g.,  $R_{p,q}$  is a head square colored in green with a star in the middle. The family of  $R_{p,q}$  is highlighted in green. Squares in each family are marked with the same shape of that family. Multiple smaller size shapes within a square implies that the square belongs to family of the corresponding multiple head squares. Observe that a head square does not belong to the family of any other head square. (Color figure online)

Our algorithm initially selects one node from each non-empty square. By selecting additional nodes from the family of each head square, we find a local connected dominating set for each family. We show that this local connections are enough to construct a connected dominating set for the underlying network.

### 3.2 The Algorithm

The algorithm is executed in 44 rounds. In the second round, at most one node from each square is selected as a member of the dominating set. The next 42 rounds are dedicated for selecting a set of additional nodes for connectivity. Every node  $v$  maintains a variable  $dom(v)$  that is initially set to zero. The details of the algorithm is as follows.

1. In round 1, a node  $v \in R_{i,j}$  for some  $i, j$  transmits a message  $\langle M, (x, y) \rangle$  to all its neighbors where the coordinates of  $v$  are  $(x, y)$ .



**Fig. 3.** (a) Family of the square  $R_{x,y}$ : Any unit disk whose center belongs to  $R_{x,y}$ , can not have intersection with any unit disk whose center does not belong to any of these 21 squares. We can see the unit disks centering in  $R_{x,y}$  lie entirely within this family. (b) Nodes selected in the family of head square  $R_{p,q}$  in round 2 are marked in large orange circles. The edges and additional nodes are obtained by replacing the edges of the spanning tree by the corresponding special paths. As an example, there was an edge  $u'w'$  in the spanning tree which is replaced by the special path  $w' - v' - u'$ . (Color figure online)

2. Let  $Z_1(v)$  be the set of nodes from which  $v$  received a message in round 1. Let  $Z'(v) \subseteq Z_1(v)$  be the set of nodes that belongs to  $R_{i,j}$ . If  $v$  is the smallest  $id$  node among  $Z'(v)$  then it sets  $dom(v) = 1$  in round 2.
3. In each round  $\ell$ , for  $3 \leq \ell \leq 43$ , the node  $v$  transmits a message  $< M, dom(v), Z_{\ell-2}(v) >$  to all its neighbors. Then it computes  $Z_{\ell-1}(v)$  as  $\cup_{u \in Z_1(v)} Z_{\ell-2}(u)$ .
4. For each  $(p, q) \in C(R_{i,j})$ , let  $S(p, q)$  be the set of nodes  $u$  in  $Z_{42}(v)$  such that  $u \in \mathfrak{F}(p, q)$  and the node  $u$  set  $dom(u) = 1$  in round 2. The node  $v$  constructs a graph  $G(p, q) = (V(p, q), E(p, q))$  as follows:
  - For each node  $u \in S(p, q)$ , take a vertex  $u(p, q)$  in  $V(p, q)$ .
  - Let  $\mathfrak{P}(u, w)$  be the set of paths of length at most 3 in  $G$  between the nodes  $u \in S(p, q)$  and  $w \in S(p, q)$  whose internal nodes are in  $R_{i,j} \cup R_{i',j'}$ , where  $u \in R_{i,j}$  and  $w \in R_{i',j'}$ . A path  $\mathcal{P} \in \mathfrak{P}(u, w)$  between those two nodes  $u$  and  $w$  is said to be the *special path* if the length of  $\mathcal{P}$  is  $dist(u, w)$  and it is lexicographically shortest from the node with smaller  $id$  (between  $u$  and  $w$ ) to the node with larger  $id$  (between  $u$  and  $w$ ). For any two vertices  $u(p, q), w(p, q) \in V(p, q)$ , add an edge between  $u(p, q), w(p, q)$  in  $E(p, q)$  if and only if  $\mathfrak{P}(u, w)$  is nonempty.

The node  $v$  computes a spanning tree  $\mathcal{T}(p, q)$  using some deterministic algorithm corresponding to each connected component  $\mathcal{C}(p, q)$  of  $G(p, q)$ . For each edge  $(u(p, q), w(p, q))$  in  $\mathcal{T}(p, q)$ ,  $v$  replaces this edge by the special path joining  $u$  and  $w$  in  $G^2$  (see Fig. 3b). If  $v$  is a vertex that lies on the special path, it sets  $dom(v) = 1$ .

The connected dominating set returned by the algorithm is the set of nodes  $\mathcal{U}$  such that for each  $u \in \mathcal{U}$ ,  $dom(u) = 1$ . We prove the correctness of the proposed algorithm and analyze its approximation factor in the next subsection.

<sup>2</sup> Note that  $(u(p, q), w(p, q)) \in \mathcal{T}(p, q)$  implies  $(u(p, q), w(p, q)) \in E(p, q)$ . This implies  $\mathfrak{P}(u, w)$  is non empty and the special path joining  $u$  and  $w$  in  $G$  exists.

### 3.3 Analysis

Let  $\mathcal{U} = \{v : \text{dom}(v) = 1\}$ . The following lemmas will help to prove that  $\mathcal{U}$  is a connected dominating set of  $G$ .

**Lemma 1.** *The set  $\mathcal{U}$  is a dominating set of  $G$ .*

*Proof.* Consider a node  $u \in G$ . If  $u \in \mathcal{U}$ , then we are done. Suppose that  $u \notin \mathcal{U}$ . Let  $R_{i,j}$  be the square such that  $u \in R_{i,j}$ . In round 2 of the algorithm, one node  $u'$  from  $R_{i,j}$  sets  $\text{dom}(u') = 1$ . Since the sides of the square  $R_{i,j}$  are of length  $\frac{1}{\sqrt{2}}$ , any two nodes in  $R_{i,j}$  are at most one distance apart and hence  $u'$  is a neighbor of  $u$ . This proves the lemma.  $\square$

**Lemma 2.** *Let  $R_{p,q}$  be a head square and  $u, u'$  be two nodes in  $\mathfrak{F}(R_{p,q})$ . If  $u$  and  $u'$  are connected in the subgraph induced by the nodes in  $\mathfrak{F}(R_{p,q})$ , then  $\text{dist}(u, u') \leq 41$ .*

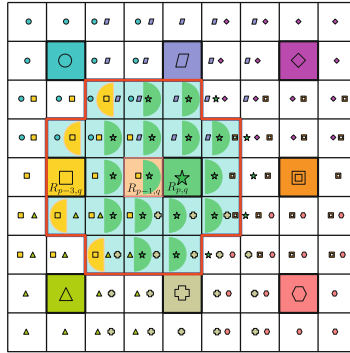
*Proof.* We prove this statement by contradiction. Suppose that there exists two nodes  $w, w' \in \mathfrak{F}(R_{p,q})$  such that the shortest path between  $w$  and  $w'$  is at least 42 in the graph induced by the nodes in  $\mathfrak{F}(R_{p,q})$ , i.e., the shortest path from  $w$  and  $w'$  contains 43 nodes including  $w$  and  $w'$  from  $\mathfrak{F}(R_{p,q})$ . Since there are 21 squares in  $\mathfrak{F}(R_{p,q})$ , by the Pigeonhole principle, there must exist 3 nodes in this path that lie in the same square. This leads to a contradiction as the diagonal of each square is 1 and hence any 3 nodes in the same square must form a cycle.  $\square$

**Lemma 3.** *For any two nodes  $u, v \in G$ , if  $u$  and  $v$  are neighbors, then there exists some head square  $R_{p,q}$  such that  $u \in \mathfrak{F}(R_{p,q})$  and  $v \in \mathfrak{F}(R_{p,q})$ .*

*Proof.* We first assume that  $u \in R_{p,q}$ , where  $R_{p,q}$  is a head square. This case is trivial, since  $\text{dist}(u, v) \leq 1$ , therefore,  $v \in \mathfrak{F}(R_{p,q})$ . Hence,  $u$  and  $v$  belongs to the same head square.

Next, suppose that  $u$  is in some square in  $\mathfrak{F}(R_{p,q}) \setminus \{R_{p,q}\}$ , where  $R_{p,q}$  is a head square. Define a set  $T_{p,q}$ , where  $T_{p,q} = \{R_{x,y} : (|x - p| \leq 1 \text{ and } |y - q| \leq 1)\}$ . So,  $T_{p,q} \subset \mathfrak{F}(R_{p,q})$ . Depending on in which square  $u$  belongs, we categorize all possibilities in 4 cases. The first 3 cases consider that  $u$  belongs to different squares of  $T_{p,q}$ , whereas, the fourth case considers that  $u$  belongs to  $\mathfrak{F}(R_{p,q}) \setminus T_{p,q}$ .

- **Case 1** [ $u \in R_{p-1,q}$ ]: Partition the set  $\mathfrak{F}(R_{p-1,q})$  into two disjoint sets  $A$  and  $B$ , where  $A = \mathfrak{F}(R_{p-1,q}) \setminus \mathfrak{F}(R_{p,q})$  and  $B = \mathfrak{F}(R_{p-1,q}) \cap \mathfrak{F}(R_{p,q})$ . If the node  $v$  is in some square in  $B$ , then both  $u$  and  $v$  belong to some squares of the family of head square  $\mathfrak{F}(R_{p,q})$ . Suppose that  $v$  is in some square  $R_{x,y}$  of the set  $A$ . Since  $R_{x,y} \in \mathfrak{F}(R_{p-1,q})$  and  $R_{x,y} \notin \mathfrak{F}(R_{p,q})$ , therefore,  $A = \{R_{p-3,q}, R_{p-3,q+1}, R_{p-3,q-1}, R_{p-2,q+2}, R_{p-2,q-2}\}$ . For each possible values of  $x, y$  such that  $R_{x,y} \in A$ ,  $|p - 3 - x| \leq 1$  and  $|q - y| \leq 2$  holds. Hence, as per the definition of the family of a square, each of the squares in  $A$  belongs to  $\mathfrak{F}(R_{p-3,q})$ . So the squares in  $A$  and  $R_{p-1,q}$  belong to the family of the head square  $R_{p-3,q}$  (See Fig. 4).



**Fig. 4. Case 1 of Lemma 3:** The square  $R_{p-1,q}$  is part of family of 2 head squares  $R_{p,q}$ , and  $R_{p-3,q}$ . The family of  $R_{p-1,q}$  is bounded by red outline. The squares in the set  $A$  are marked by yellow half-circles along with the yellow head square  $R_{p-3,q}$ . The node  $u$  and any neighbor of  $u$  in  $A$  shares the common head square  $R_{p-3,q}$ . The squares in  $B$  are marked by green half-circles along with the green head square  $R_{p,q}$ . The node  $u$  and any neighbor of  $u$  in  $B$  shares the common head square  $R_{p,q}$ . (Color figure online)

- **Case 2** [ $u \in R_{p-1,q+1}$ ]: Partition the set  $\mathfrak{F}(R_{p-1,q+1})$  into three disjoint sets  $A$ ,  $B$ , and  $C$ , where  $A = \mathfrak{F}(R_{p-1,q+1}) \cap \mathfrak{F}(R_{p,q})$ ,  $B = (\mathfrak{F}(R_{p-1,q+1}) \setminus A) \cap \mathfrak{F}(R_{p,q+3})$ , and  $C = (\mathfrak{F}(R_{p-1,q+1}) \setminus (A \cup B)) \cup \mathfrak{F}(R_{p-3,q})$ . If the node  $v$  is in some square  $R_{x,y}$  of  $A$ , then both  $R_{x,y}$  and  $R_{p-1,q+1}$  are in the family of the head square  $R_{p,q}$ . If  $R_{x,y}$  is in  $B$ , then both  $R_{x,y}$  and  $R_{p-1,q+1}$  are in the family of the head square  $R_{p,q+3}$ . If  $R_{x,y}$  is in  $C$ , both  $R_{x,y}$  and  $R_{p-1,q+1}$  are in the family of the head square  $R_{p-3,q}$ .
- **Case 3** [ $u \in R_{x,y}$ , where  $R_{x,y} \in T_{p,q} \setminus \{R_{p-1,q}, R_{p-1,q+1}\}$ ]: The squares  $R_{p+1,q}$ ,  $R_{p,q+1}$ , and  $R_{p,q-1} \in T_{p,q}$  are symmetrical neighboring squares of  $R_{p,q}$  w.r.t.  $R_{p-1,q}$ . Hence similar arguments of case 1 can be used to prove the statement of the lemma for these squares. Similarly,  $R_{p-1,q-1}$ ,  $R_{p+1,q+1}$  and  $R_{p+1,q-1}$  are symmetrical neighboring squares of  $R_{p,q}$  w.r.t.  $R_{p-1,q+1}$ . Hence similar arguments of case 1 can be used to prove the statement of the lemma for these squares.
- **Case 4** [ $u \in R_{x,y}$ , where  $R_{x,y} \in \mathfrak{F}(R_{p,q}) \setminus T_{p,q}$ ]: The following subcases exhaust all possibilities (referring the structure of  $\mathfrak{F}(R_{p,q})$ ).
  - $u \in R_{x,y}$  such that  $x = p - 2$ : These squares belong to  $T_{p-3,q}$ .
  - $u \in R_{x,y}$  such that  $x = p + 2$ : These squares belong to  $T_{p+3,q}$ .
  - $u \in R_{x,y}$  such that  $y = q + 2$ : These squares belong to  $T_{p,q+3}$  or  $T_{p-3,q+3}$ .
  - $u \in R_{x,y}$  such that  $y = q - 2$ : These squares belong to  $T_{p,q-3}$  or  $T_{p-3,q-3}$ .
 Hence the proof of the lemma will hold as it holds for the squares in  $T_{p,q}$  for any arbitrary head square  $R_{p,q}$ . □

**Theorem 3.** *The graph induced by the nodes in  $\mathcal{U}$  is connected.*

*Proof.* It is sufficient to prove that there exists a path between two nodes  $u$  and  $v$  in  $\mathcal{U}$  such that the path contains only nodes in  $\mathcal{U}$ .

Since the graph  $G$  is connected, there exists a path  $P$  between  $u$  and  $v$  in  $G$ . Let  $P = (u =)v_0 - v_1 - \dots - v_t(= v)$ . Let  $u$  belong to a square in  $\mathfrak{F}(R_{p,q})$  and let  $v_j$  be the first node in the path  $P$  such that  $v_j$  belongs to a square in  $\mathfrak{F}(R_{p,q})$  and  $v_{j+1}$  does not belong to any square of  $\mathfrak{F}(R_{p,q})$ . Let  $v_j \in R_{a,b}$  and  $v_{j+1} \in R_{x,y}$ . A node  $u_j \in R_{a,b}$  and  $u_{j+1} \in R_{x,y}$  have set  $dom = 1$  in round 2 of the algorithm. Since  $v_j$  and  $v_{j+1}$  are neighbors, by Lemma 3,  $R_{a,b}$  and  $R_{x,y}$  belongs to  $\mathfrak{F}(R_{p',q'})$ , for some head square  $R_{p',q'}$ . Since  $dist(u_j, u_{j+1})$  is at most 3 in  $G$  and one such distance 3 path is  $u_j - v_j - v_{j+1} - u_{j+1}$  where  $v_j, v_{j+1} \in R_{a,b} \cup R_{x,y}$ . Therefore, in the graph  $G(p', q')$ , they are connected by at least one edge (Step 4 of the algorithm) and hence, they are in the same connected component  $\mathcal{C}$  of  $G(p', q')$ . By Lemma 2, any two nodes in the same connected component of  $G(p', q')$  are at most 41 distance apart in  $G$ . Therefore, in Step 4 of the algorithm, all the nodes of  $\mathcal{C}$  compute the same graph  $G(p', q')$  and hence, compute the same spanning tree. Therefore, the nodes  $u_j$  and  $u_{j+1}$  are connected by a path consisting of nodes in  $\mathcal{U}$ . Also, since  $u$  and  $u_j$  are in the same connected component of  $G(p, q)$ , there exists a path between  $u$  and  $u_j$  consisting of nodes in  $\mathcal{U}$ . This proves that there exists a path between  $u$  and  $u_{j+1}$  consisting of nodes in  $\mathcal{U}$ .

Using the above argument recursively, we can show that there exists a path between  $u_{j+1}$  to  $v$  consisting of nodes in  $\mathcal{U}$ . Hence the theorem follows.  $\square$

**Theorem 4.** *The proposed algorithm returns a 441-approximation connected dominating set in 44 rounds using  $O(n)$  messages where  $n$  is the number of nodes.*

*Proof.* Let  $u$  be a node selected in an optimal connected dominating set of  $G$ . Further, let  $u \in R_{x,y}$ . Note that  $u$  alone may dominate all the nodes belong to the squares in  $\mathfrak{F}(R_{x,y})$ . We show that according to our algorithm, at most 441 nodes are selected from  $\mathfrak{F}(R_{x,y})$ .

**Claim:** From any square  $R_{a,b}$  at most 21 nodes are selected in  $\mathcal{U}$  by the end of our algorithm.

To prove the above claim consider the family  $\mathfrak{F}(R_{a,b})$ . Let  $v \in R_{a,b}$  be the node that set  $dom(v) = 1$  in round 2. Let  $\mathfrak{S}$  be the set of nodes selected in  $\mathcal{U}$  from the remaining 20 squares of  $\mathfrak{F}(R_{a,b})$  in round 2. Since one node per square is added in  $\mathcal{U}$  in round 2,  $|\mathfrak{S}| \leq 20$ . Therefore, there is at most 20 special paths present between  $v$  and the nodes in  $\mathfrak{S}$ . In round 44 of our algorithm, one additional node from  $R_{a,b}$  corresponding to each of these 20 special paths may be added in  $\mathcal{U}$ . Hence, in total,  $\mathcal{U}$  contains at most 21 nodes from  $R_{a,b}$ .

As there are 21 squares in  $\mathfrak{F}(R_{x,y})$ , using the above claim we can say, our algorithm may choose at most  $21 \times 21 = 441$  nodes from  $\mathfrak{F}(R_{x,y})$ . As  $u \in R_{x,y}$  is already selected in the optimal solution, we conclude our theorem.

It is clear from the algorithm that it runs for no more than 44 rounds. Each node broadcasts only constant number of times and hence the message complexity becomes  $O(n)$ .  $\square$



## 4 Conclusion

We study the minimum dominating set and minimum connected dominated set problems in a distributed setting on axis-parallel unit square graphs and unit disk graphs. We provide constant rounds distributed algorithms using linear number of messages for these classes of graphs and show that these algorithms are constant factor approximation algorithms. Similar problems on geometric intersection graphs where the nodes have weights would be an interesting direction for further study.

## References

1. Alzoubi, K.M., Wan, P., Frieder, O.: Message-optimal connected dominating sets in mobile ad hoc networks. In: *MobiHoc*, pp. 157–164 (2002)
2. Ambühl, C., Erlebach, T., Mihalák, M., Nunkesser, M.: Constant-factor approximation for minimum-weight (connected) dominating sets in unit disk graphs. In: *APPROX-RANDOM*, vol. 4110, pp. 3–14 (2006)
3. Cheng, X., Ding, M., Du, D.H., Jia, X.: Virtual backbone construction in multihop ad hoc wireless networks. *Wirel. Commun. Mob. Comput.* **6**, 183–190 (2006)
4. Cheng, X., Huang, X., Li, D., Wu, W., Du, D.: A polynomial-time approximation scheme for the minimum-connected dominating set in ad hoc wireless networks. *Networks* **42**(4), 202–208 (2003)
5. Clark, B.N., Colbourn, C.J., Johnson, D.S.: Unit disk graphs. *Discret. Math.* **86**(1–3), 165–177 (1990)
6. Das, B., Bharghavan, V.: Routing in ad-hoc networks using minimum connected dominating sets. In: *ICC*, pp. 376–380 (1997)
7. Das, B., Sivakumar, R., Bharghavan, V.: Routing in ad hoc networks using a spine. In: *ICCCN*, pp. 34–41 (1997)
8. Funke, S., Kesselman, A., Meyer, U., Segal, M.: A simple improved distributed algorithm for minimum CDS in unit disk graphs. *ACM Trans. Sens. Networks* **2**(3), 444–453 (2006)
9. Gao, B., Yang, Y., Ma, H.: A new distributed approximation algorithm for constructing minimum connected dominating set in wireless ad hoc networks. *Int. J. Commun. Syst.* **18**(8), 743–762 (2005)
10. III, H.B.H., Marathe, M.V., Radhakrishnan, V., Ravi, S.S., Rosenkrantz, D.J., Stearns, R.E.: NC-approximation schemes for NP- and PSPACE-hard problems for geometric graphs. *J. Algorithms* **26**(2), 238–274 (1998)
11. Jallu, R.K., Prasad, P.R., Das, G.K.: Distributed construction of connected dominating set in unit disk graphs. *J. Parallel Distrib. Comput.* **104**, 159–166 (2017)
12. Lichtenstein, D.: Planar formulae and their uses. *SIAM J. Comput.* **11**(2), 329–343 (1982)
13. Marathe, M.V., Breu, H., III, H.B.H., Ravi, S.S., Rosenkrantz, D.J.: Simple heuristics for unit disk graphs. *Networks* **25**(2), 59–68 (1995)
14. Min, M., Du, H., Jia, X., Huang, C.X., Huang, S.C., Wu, W.: Improving construction for connected dominating set with steiner tree in wireless sensor networks. *J. Glob. Optim.* **35**(1), 111–119 (2006)
15. Mohanty, J.P., Mandal, C.A., Reade, C.: Distributed construction of minimum connected dominating set in wireless sensor network using two-hop information. *Comput. Netw.* **123**, 137–152 (2017)

16. Peleg, D.: Distributed computing: a locality-sensitive approach. Soc. Industr. Appl. Math. (2000)
17. Wan, P., Alzoubi, K.M., Frieder, O.: Distributed construction of connected dominating set in wireless ad hoc networks. In: INFOCOM, pp. 1597–1604 (2002)
18. Wu, J., Li, H.: On calculating connected dominating set for efficient routing in ad hoc wireless networks. In: DIAL-M, pp. 7–14 (1999)
19. Wu, W., Du, H., Jia, X., Li, Y., Huang, S.C.: Minimum connected dominating sets and maximal independent sets in unit disk graphs. Theor. Comput. Sci. **352**(1–3), 1–7 (2006)



# An Inventory System Optimization for Solving Joint Pricing and Ordering Problem with Trapezoidal Demand and Partial Backlogged Shortages in a Limited Sales Period

Chunming Xu<sup>1(✉)</sup>, Mingfei Bai<sup>2</sup>, Qiyue Wang<sup>2</sup>, and Yiwei Wang<sup>2</sup>

<sup>1</sup> Institute of Operations Research and Systems Engineering, College of Science, Tianjin University of Technology, Tianjin 300384, People's Republic of China  
chunmingxu@tjut.edu.cn

<sup>2</sup> School of Electrical and Electronic Engineering, Tianjin University of Technology, Tianjin 300384, People's Republic of China

**Abstract.** This study investigates a joint pricing and ordering model with deteriorating items and partial backlogged shortages under trapezoidal demand. The optimization model aiming at maximizing profit performance of inventory system is subject to a limited sales period and an allowable price range, simultaneously. The existence and uniqueness of the optimal solution to the model is further given by applying differential calculus. The easy-to-use line search algorithm is designed to determine the optimal replenishment strategies including ordering quantity, the maximum inventory level, selling price, and shortage time point. Finally, numerical examples are presented to demonstrate the proposed model.

**Keywords:** Inventory · Trapezoidal demand · Pricing · Deteriorating items · Partial backlogged shortages

## 1 Introduction

With the further advancement of economic globalization, inventory control of goods has been paid more and more attention. However, too much inventory of goods will lead to a large amount of waste and occupation of capital, affecting the core competitiveness of enterprises. Therefore, it is of great practical value to study how to manage and control the inventory of goods, rationally.

In the traditional inventory problem with the time-varying demand, the demand for goods in the market is usually considered to decrease or increase over time. However, with the development of the times, the speed of product renewal is getting faster and faster, some electronic products such as clothing and electronics take only a few months from appearance to elimination in the

market. Therefore, the demand for products during sales period may not be a single rise or fall. Especially, the demand for seasonal or fashionable products initially increases over time as potential consumers are attracted by style and quality, then tends to stabilize as the type of product is accepted by the market, and finally decreases over time. It was found by Micheal, Rochford, and Wotruba (2003). Subsequently, Cheng and Wang (2009) investigated an inventory model for deteriorating items with trapezoidal type demand rate. Afterwards, Lin (2013) and Glock and Grosse (2015) pointed that most products basically follow this trapezoidal demand feature in their life cycle. In recent years, the trapezoidal time-varying inventory models have been widely studied (Cheng, Zhang, and Wang, 2011; Lin, 2013; Shah, Shah, and Patel, 2013; Singh, Vaish, and Singh, 2010; Uthayakumar and Rameswari, 2012; Xu, Zhao, Min, and Hao, 2021).

Another important aspect is the consideration of the shortage of goods. In the early study, the out of stock status was not considered and replenishment would occur immediately when the goods were sold out, which could be seen in the literature (Dye, Chang, and Teng, 2006; Teng, Chern, and Yang, 1997; Yang, Teng, and Chern, 2001). However, this is obviously not an optimal inventory replenishment policy. Researchers gradually realize that due to the increase of inventory cost and the impact of commodity deterioration, which will lead to large occupation of operating costs in the inventory system. In later studies, out of stock states have been described as an integral part of a cycle (Abad, 2008; Dye, 2007; Zhou, Lau, and Yang, 2003). However, in the state of shortage, either the customer is presumed to be completely lost or the customer is expected to wait in most literature. Obviously, there are imperfections in both of considerations. In fact, during the period of shortage, customers will wait for some time. The length of this period is related to the customer's loyalty to the product, the urgency of customer demand, whether the product can be replaced, and so on. Therefore, the backlogging rate during shortage period depends on the waiting time for the next replenishment. Dye (2007) developed an inventory model with a time-varying backlogging rate. Recently, most of the studies have centered on this backlogging rate (Ghosh, Khanra, Chaudhuri, 2011; Khan, Shaikh, Panda, Konstantaras, and Taleizadeh, 2019; Lashgari, Taleizadeh, and Sadjadi, 2018; Salehi, Taleizadeh, and Tavakkoli-Moghaddam, 2016; Taleizadeh, 2018; Taleizadeh, Khanbaglo, and Eduardo, 2016; Xu, Bisi, and Dada, 2017).

However, existing studies often assumed that the customer demand is characterized as a function of time, stock level or selling price, separately (Khan, Shaikh, Panda, Konstantaras, and Taleizadeh, 2019; Xu, Zhao, Min, and Hao, 2021; Dye, Chang, and Teng, 2006; Lashgari, Taleizadeh, and Sadjadi, 2018). In the actual retail operation, the time and the selling price ought to be investigated jointly. The reason behind this observation is that the demand is closely related to the market stage of the product and the selling price, namely, the customer demand may vary with the time, and meanwhile it may also vary when the selling price decreases or increases. As a result, we will study a new inventory system for solving joint pricing and ordering problem with trapezoidal demand

under a fixed replenishment period. We generalize the metamorphism function to apply to more complex cases. Shortages are allowed and unsatisfied demands are assumed to be partially backlogged during the stock-out period.

The rest of this paper is organized as follows. In Sect. 2, the assumptions and notations used throughout this study are introduced. In Sect. 3, the mathematical model to maximize the total revenue is developed. Besides, we discuss the optimal solution to the model and verify its existence and uniqueness. Section 4 gives a simple algorithm to find the optimal inventory replenishment strategy. In Sect. 5, numerical examples are given to illustrate the proposed model. In Sect. 6, some conclusions are summarized and future directions are given.

## 2 Model Description

Consider a continuous review retail system, where the retailer sells products kept in the system to the end customer in a finite inventory planning horizon. The retail replenishment for products is instantaneous and the lead time is zero. When products from the upstream manufacturer enter the retail system, they are exhausted because of the deterioration and customer demand until the inventory level is zero. Retail system allows shortages. The retailer is able to forecast customer demand and determine the initial ordering quantity according to the demand and shortages. Once shortages happen, backlogged demand is satisfied at the end of the inventory cycle.

The assumptions are used throughout the paper.

1. The replenishment rate is infinite and the lead time is zero.
2. Shortages are allowed. The fraction of backlogging rate is described as  $\beta(t) = e^{-\delta t}$ , where  $\delta > 0$ ,  $t$  is the waiting time in the time of stock-out. The backlogging rate function has already been widely used in the literature (Abad, 1996).
3. Demand is related to time and price, so the demand function can be described as  $D(t) \cdot D(p)$ . Among them,

$$D(t) = \begin{cases} a_1 + b_1t, & 0 \leq t \leq \mu_1; \\ D_0, & \mu_1 \leq t \leq \mu_2; \\ a_2 - b_2t, & \mu_2 \leq t \leq T \leq \frac{a_2}{b_2}, \end{cases}$$

where  $\mu_1$  is the time point when demand grows linearly to constant demand, and  $\mu_2$  is the point at which the demand goes from constant to linearly decreasing (see, Fig. 1.), and  $D(p)$  is a function of selling price  $p$ .

4. It is necessary to make some assumptions based on the actual situation,  $D(p)$  satisfies that  $D(p)'' \geq 0$  and  $2D'(p) + pD''(p) < 0$  (Khan et al. (2019) and Maihami (2012)).

5. Before the inventory system starts, the deterioration of transporting goods and the logistics cost from the manufacturer to the system are ignored.

6. To keep up with the real situation, the interval of  $p$  is denoted as  $p \in [p_{min}, p_{max}]$ .

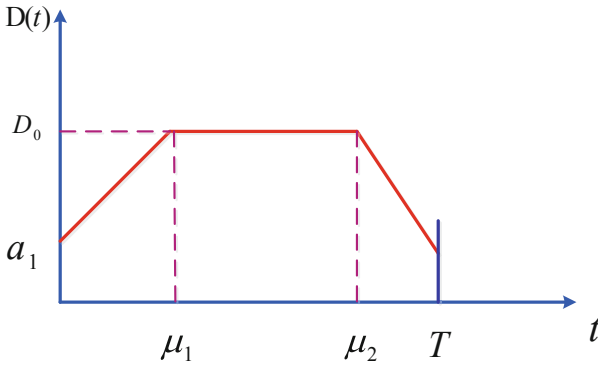


Fig. 1. Trapezoidal time-varying demand description

The following notations including model parameters and domain parameters are listed.

$I_i(t)$  is the inventory level for case  $i$ ,  $i = 1, 2, 3, 4$ ;  $T$  is the fixed length of the replenishment cycle time;  $\theta(t)$  is the nonnegative function of the deterioration rate;  $t_1$  is the length of time that an inventory shortage did not occur (**decision variable**);  $t_1^*$  is the optimal point that an inventory shortage did not occur;  $A_0$  is the fixed ordering cost each order;  $p$  is the selling price each unit (**decision variable**);  $p_{min}$  is the lower bound of the allowable price;  $p_{max}$  is the upper bound of the allowable price;  $p^*$  is the optimal selling price each unit;  $C$  is the constant purchasing cost each unit;  $C_1$  is the cost of per deteriorated item;  $C_2$  is the inventory holding cost each unit each unit of time;  $C_3$  is the shortage cost each unit each unit of time;  $C_4$  is the cost of sales each unit lost;  $S_i$  is the maximum inventory level for case  $i$ , where  $i = 1, 2, 3$ ;  $S^*$  is the optimal maximum inventory level;  $Q_i$  is the order quantity level for case  $i$ , where  $i = 1, 2, 3$ ;  $Q^*$  is the optimal order quantity.  $ATP_i(p, t_1)$  is the total profit each unit time of this inventory system for case  $D_i$ , where  $D_1 = \{(t_1, p) | 0 \leq t_1 \leq \mu_1, p_{min} \leq p \leq p_{max}\}$ ,  $D_2 = \{(t_1, p) | \mu_1 \leq t_1 \leq \mu_2, p_{min} \leq p \leq p_{max}\}$ , and  $D_3 = \{(t_1, p) | \mu_2 \leq t_1 \leq T, p_{min} \leq p \leq p_{max}\}$ ;  $ATP(p, t_1)$  (**objective function**) is the total profit each unit time of this inventory system for case  $D$  (**feasible region**), where  $D = \bigcup_{i=1}^3 D_i$ ;  $ATP^*$  is the maximum value of  $ATP(p, t_1)$ , that is  $ATP^* = ATP(p^*, t_1^*)$ .

### 3 Model and Model Analysis

Based on the assumptions and notations mentioned above, the initial replenishment occurs at time  $t = 0$ . During the time interval  $[0, t_1]$ , the inventory level becomes lower due to demand and deterioration. Inventory levels will fall to zero when  $t = t_1$ . During the time interval  $[t_1, T]$ , the demand is partially backlogged and the sales losses occurred. The backordered items will be replenished at  $t = T$ .

So the behavior of the inventory system at any time  $t$  can be described by the differential equations as follows.

$$\frac{dI(t)}{dt} = \begin{cases} -\theta(t)I(t) - D(t)D(p), & 0 \leq t \leq t_1; \\ -D(t)D(p)e^{-\delta(T-t)}, & t_1 \leq t \leq T. \end{cases} \tag{1}$$

Considering possible values of  $\mu_1, \mu_2, t_1, p,$  and  $T,$  three different inventory cases are explored as follows.

### 3.1 Case with $D_1$

In this case, inventory depletion occurs in  $[0, t_1]$  due to both the demand  $(a_1 + b_1t)d(p)$  and the deterioration  $\theta(t);$  In  $[t_1, T],$  there is no deteriorating phenomenon in the system, the inventory depletion happens due to both the demand and the partial backlogging. Hence,  $I(t)$  during  $[0, T]$  can be described by

$$\begin{cases} \frac{dI_1(t)}{dt} = -\theta(t)I_1(t) - (a_1 + b_1t)D(p), & 0 \leq t \leq t_1; \\ \frac{dI_2(t)}{dt} = -(a_1 + b_1t)D(p)e^{-\delta(T-t)}, & t_1 \leq t \leq \mu_1; \\ \frac{dI_3(t)}{dt} = -D_0D(p)e^{-\delta(T-t)}, & \mu_1 \leq t \leq \mu_2; \\ \frac{dI_4(t)}{dt} = -(a_2 - b_2t)D(p)e^{-\delta(T-t)}, & \mu_2 \leq t \leq T. \end{cases} \tag{2}$$

Solving Eq.(2) with the boundary conditions  $I(t_1) = 0, I(t_1) = 0, I(\mu_1^-) = I(\mu_1^+),$  and  $I(\mu_2^-) = I(\mu_2^+),$  we have

$$\begin{cases} I_1(t) = -D(p)e^{-\int_0^t \theta(x)dx} \int_{t_1}^t (a_1 + b_1x)e^{\int_0^x \theta(y)dy} dx, & 0 \leq t \leq t_1; \\ I_2(t) = -\frac{D(p)}{\delta^2} e^{-\delta T} \left[ e^{\delta t} (a_1\delta + b_1\delta t - b_1) - e^{\delta t_1} (a_1\delta + b_1\delta t_1 - b_1) \right], & t_1 \leq t \leq \mu_1; \\ I_3(t) = -\frac{D_0D(p)}{\delta} e^{-\delta T} (e^{\delta t} - e^{\delta \mu_1}) + I_2(\mu_1), & \mu_1 \leq t \leq \mu_2; \\ I_4(t) = -\frac{D(p)}{\delta^2} e^{-\delta T} \left[ e^{\delta t} (a_2\delta - b_2\delta t + b_2) - e^{\delta \mu_2} (a_2\delta - b_2\delta \mu_2 + b_2) \right] + I_3(\mu_2), & \mu_2 \leq t \leq T. \end{cases} \tag{3}$$

From Eq.(3), the maximum inventory level can be calculated as

$$S_1 = I_1(0) = D(p) \int_0^{t_1} (a_1 + b_1x)e^{\int_0^x \theta(y)dy} dx, \tag{4}$$

and the order quantity is

$$Q_1 = S_1 - I_4(T). \tag{5}$$

To sum up, from Eqs.(2), (3), (4), and (5), the related cost and total revenue in  $[0, T]$  can be calculated as

- (a) The ordering cost  $A_0;$
- (b) The cost of deteriorated item  $D_{T1} = C_1 \left[ S_1 - \int_0^{t_1} (a_1 + b_1t)D(p)dt \right];$
- (c) The inventory holding cost  $H_{T1} = C_2 \int_0^{t_1} I_1(t)dt;$
- (d) The cost of shortage due to backlog:  $B_{T1} = -C_3 \left[ \int_{t_1}^{\mu_1} I_2(t)dt + \int_{\mu_1}^{\mu_2} I_3(t)dt + \int_{\mu_2}^T I_4(t)dt \right];$

(e) The opportunity cost caused by the lost sales  $O_{T1} = C_4 D(p) \left\{ \int_{t_1}^{\mu_1} (a_1 + b_1 t) [1 - e^{-\delta(T-t)}] dt + \int_{\mu_1}^{\mu_2} D_0 [1 - e^{-\delta(T-t)}] dt + \int_{\mu_2}^T (a_2 - b_2 t) [1 - e^{-\delta(T-t)}] dt \right\}$ ;

(f) The purchase cost  $E_{T1} = CQ_1$ ;

(g) The sales revenue  $F_1 = p \left[ \int_0^{t_1} D(p) (a_1 + b_1 t) dt - I_4(T) \right]$ .

Therefore, the total profit per unit time (denoted by  $ATP_1(p, t_1)$ ) is given by

$$\begin{aligned}
 ATP_1(p, t_1) &= \frac{1}{T} (F_1 - H_{T1} - B_{T1} - O_{T1} - D_{T1} - E_{T1} - A_0) \\
 &= \frac{1}{T} \left\{ \underbrace{p \left[ \int_0^{t_1} D(p) (a_1 + b_1 t) dt - I_4(T) \right]}_{\text{the total sales revenue}} \right. \\
 &\quad - \underbrace{C_1 \left[ S_1 - \int_0^{t_1} (a_1 + b_1 t) D(p) dt \right]}_{\text{the cost of deteriorated item}} \\
 &\quad - \underbrace{C_2 \int_0^{t_1} I_1(t) dt}_{\text{the holding cost}} + \underbrace{C_3 \left[ \int_{t_1}^{\mu_1} I_2(t) dt + \int_{\mu_1}^{\mu_2} I_3(t) dt + \int_{\mu_2}^T I_4(t) dt \right]}_{\text{the cost of shortage}} \\
 &\quad - \underbrace{C_4 D(p) \left[ \int_{t_1}^{\mu_1} (a_1 + b_1 t) [1 - e^{-\delta(T-t)}] dt + \int_{\mu_1}^{\mu_2} D_0 [1 - e^{-\delta(T-t)}] dt \right.}_{\text{the opportunity cost due to lost sales}} \\
 &\quad \left. + \int_{\mu_2}^T (a_2 - b_2 t) [1 - e^{-\delta(T-t)}] dt \right\} - \underbrace{A_0}_{\text{the ordering cost}} \quad (6)
 \end{aligned}$$

### 3.2 The Other Two Cases

Similar to case with  $D_1$ , the total average profits for cases with  $D_2$  and  $D_3$  are obtained, respectively, by

$$\begin{aligned}
 ATP_2(t_1, p) &= \frac{1}{T} \left\{ p \left[ \int_0^{\mu_1} D(p) (a_1 + b_1 t) dt + \int_{\mu_1}^{t_1} D(p) D_0 dt - I_4(T) \right] \right. \\
 &\quad - C_2 \left[ \int_0^{\mu_1} I_1(t) dt + \int_{\mu_1}^{t_1} I_2(t) dt \right] + C_3 \left[ \int_{t_1}^{\mu_2} I_3(t) dt + \int_{\mu_2}^T I_4(t) dt \right] \\
 &\quad - C_4 D(p) \left\{ \int_{t_1}^{\mu_2} D_0 [1 - e^{-\delta(T-t)}] dt + \int_{\mu_2}^T (a_2 - b_2 t) [1 - e^{-\delta(T-t)}] dt \right\} \\
 &\quad \left. - C_1 \left[ S_2 - \int_0^{\mu_1} (a_1 + b_1 t) D(p) dt - \int_{\mu_1}^{t_1} D_0 D(p) dt \right] - C[S - I_4(T)] - A_0 \right\}. \quad (7)
 \end{aligned}$$



and

$$\begin{aligned}
 ATP_3(t_1, p) = & \frac{1}{T} \left\{ p \left[ \int_0^{\mu_1} D(p)(a_1 + b_1 t) dt + \int_{\mu_1}^{\mu_2} D(p) D_0 dt + \int_{\mu_2}^{t_1} D(p)(a_2 - b_2 t) dt - I_4(T) \right] \right. \\
 & - C_2 \left[ \int_0^{\mu_1} I_1(t) dt + \int_{\mu_1}^{\mu_2} I_2(t) dt + \int_{\mu_2}^{t_1} I_3(t) dt \right] + C_3 \left[ \int_{t_1}^T I_4(t) dt \right] \\
 & - C_4 D(p) \left\{ \int_{t_1}^T (a_2 - b_2 t) [1 - e^{-\delta(T-t)}] dt \right\} - C_1 \left[ S_3 - \int_0^{\mu_1} (a_1 + b_1 t) D(p) dt \right. \\
 & \left. - \int_{\mu_1}^{\mu_2} D_0 D(p) dt - \int_{\mu_2}^{t_1} (a_2 - b_2 t) D(p) dt \right] - C[S - I_4(T) - A_0] \left. \right\}. \tag{8}
 \end{aligned}$$

The formulations of the other two cases are analogous and omitted here.

Based on the above discussion, the total profit each unit time of this inventory system in the region of  $D$  is

$$ATP(t_1, p) = \begin{cases} ATP_1(t_1, p), & (t_1, p) \in D_1; \\ ATP_2(t_1, p), & (t_1, p) \in D_2; \\ ATP_3(t_1, p), & (t_1, p) \in D_3, \end{cases} \tag{9}$$

where  $ATP_1(t_1, p)$ ,  $ATP_2(t_1, p)$ , and  $ATP_3(t_1, p)$  are obtained by Eqs.(6), (7) and (8), respectively. Then, the nonlinear programming model for this system can be formulated as below:

$$\begin{aligned}
 M: \quad & \max \quad ATP(t_1, p) \\
 \text{s.t.} \quad & (t_1, p) \in D. \tag{10}
 \end{aligned}$$

To obtain the optimal solution to the model M, we have the following theorems.

**Theorem 1.** *In the model M, the first-order necessary criteria for maximizing the objective function  $AP(t_1, p)$  is equivalent to the criteria that  $f(t_1, p) = 0$  and  $g(t_1, p) = 0$ , where  $f(t_1, p)$  and  $g(t_1, p)$  can be provided by*

$$\begin{aligned}
 f(t_1, p) = & p [1 - e^{-\delta(T-t_1)}] - C_1 [e^{\int_0^{t_1} \theta(y) dy} - 1] - C_2 \left[ \int_0^{t_1} e^{-\int_0^t \theta(x) dx} e^{\int_0^{t_1} \theta(y) dy} dt \right] \\
 & + C_3 \left[ \int_{t_1}^T e^{-\delta(T-t)} dt \right] + C_4 [1 - e^{-\delta(T-t)}] - C [e^{\int_0^{t_1} \theta(y) dy} - e^{-\delta(T-t_1)}] \tag{11}
 \end{aligned}$$

and

$$g(t_1, p) = D'(p) K(t_1) + [D(p) + pD'(p)] M(t_1). \tag{12}$$

Theorem 1 implies that the first-order necessary condition for the model optimality depends not only on the unit purchasing cost and the costs incurred by storage, shortages and lost sales, but also on the trapezoidal time and the price. Next, we will explore the solutions to Eqs.(11) and (12). From Eq.(11), let  $F(t_1) = f(t_1, p)$  for any given  $p$ . The following theorem can be obtained.

**Theorem 2.**  *$F(t_1)$  is a monotonically decreasing function in  $t_1 \in [0, T]$ , and there exists a unique time point  $t_1$  satisfying  $f(t_1, p) = 0$ .*

Theorem 2 indicates that  $t_1$  is determined uniquely as a function of  $p$ . So, the function relationship between them can be described as  $t_1 = t_1(p)$ . Substituting  $t_1 = t_1(p)$  into Eq.(12) and letting  $G(p) = g(t_1(p), p)$ , the following results are obtained.

**Theorem 3.** *If  $d'(p)[\alpha(t_1) + p\gamma(t_1)] + d(p)\gamma(t_1) < 0$  holds, then  $G(p)$  is a monotonically decreasing function in  $p \in (p_{min}, p_{max})$ , where  $\alpha(t_1)$  and  $\gamma(t_1)$  can be provided by*

$$\alpha(t_1) = -e^{\int_0^{t_1} \theta(y) dy} \left[ C + C_2 \int_0^{t_1} e^{-\int_0^t \theta(x) dx} dt + C_1 \right] + e^{-\delta(T-t_1)} [C + C_3(T - t_1) - c_4] + C_1 + C_4 \tag{13}$$

and

$$\gamma(t_1) = 1 - e^{-\delta(T-t_1)}. \tag{14}$$

From Theorem 3, we have the following results.

**Theorem 4.** (1) *If  $G(p_{max}) \geq 0$ , the optimal solution is  $(t_1^*, p^*)$ , where  $p^* = p_{max}$ , and  $t_1^*$  is the solution of  $f(t_1, p) = 0$ ;*

(2) *If  $G(p_{min}) \leq 0$ , the optimal solution is  $(t_1^*, p^*)$ , where  $p^* = p_{min}$ , and  $t_1^*$  is the solution of  $f(t_1, p) = 0$ ;*

(3) *If  $G(p_{min}) > 0$  and  $G(p_{max}) < 0$ , the optimal solution is  $(t_1^*, p^*)$ , where  $(t_1^*, p^*)$  is the solution of equations  $f(t_1, p) = 0$  and  $g(t_1, p) = 0$ .*

Specially, we will list the optimal solution to the model M. From Theorem 4, we have

**Theorem 5.** *In the model M, let  $(t_1^*, p^*)$  be the optimal solution that maximizes  $ATP(t_1, p)$  in the region  $D$ , we have*

- (1) *If  $(t_1^*, p^*) \in D_1$ , then  $ATP(t_1^*, p^*) = ATP_1(t_1^*, p^*)$ ;*
- (2) *If  $(t_1^*, p^*) \in D_2$ , then  $ATP(t_1^*, p^*) = ATP_2(t_1^*, p^*)$ ;*
- (3) *If  $(t_1^*, p^*) \in D_3$ , then  $ATP(t_1^*, p^*) = ATP_3(t_1^*, p^*)$ .*

Integrate the findings of Theorem 1 to Theorem 5, a solving algorithm is formulated.

## 4 Algorithm

**Step 1.** Input all the parameters.

**Step 2.** Plug  $p = p_{min}$  in Eq.(10) to get  $\tilde{t}_1$ , where  $\tilde{t}_1$  is the solution of  $f(t_1, p_{min}) = 0$ .

**Step 3.** Plug  $p = p_{max}$  in Eq.(10) to get  $\hat{t}_1$ , where  $\hat{t}_1$  is the solution of  $f(t_1, p_{max}) = 0$ .

**Step 4.** Judge the signs of  $G(p_{min}) = G(\tilde{t}_1, p_{min})$  and  $G(p_{max}) = G(\hat{t}_1, p_{max})$ .

If  $G(p_{min}) \leq 0$ , then  $t_1^* = \tilde{t}_1, p^* = p_{min}$ , jump to **step 6**;

If  $G(p_{max}) \geq 0$ , then  $t_1^* = \hat{t}_1, p^* = p_{max}$ , jump to **step 6**;

If  $G(p_{min}) > 0$  and  $G(p_{max}) < 0$ , jump to **step 5**.

**Step 5.** Eqs.(10) and (12) are solved by Newton-Raphson method.

**Step 6.** Determine the region of  $(t_1^*, p^*)$ .

If  $(t_1^*, p^*) \in D_1$ , then  $ATP(t_1^*, p^*) = ATP_1(t_1^*, p^*)$ ;

If  $(t_1^*, p^*) \in D_2$ , then  $ATP(t_1^*, p^*) = ATP_2(t_1^*, p^*)$ ;

If  $(t_1^*, p^*) \in D_3$ , then  $ATP(t_1^*, p^*) = ATP_3(t_1^*, p^*)$ .

Then get  $S^*, Q^*, ATP^*$ .

## 5 Numerical Examples

In order to further verify the model in this paper, the following three numerical examples are given. To be specific, Example 1 shows that the optimal solution of the model exists within the feasible region  $D$ . Examples 2 and 3 imply that the optimal solution for the model occurs at the boundary point of the feasible region  $D$ .

### 5.1 Example 1

This example is based on the following data:  $A_0 = \$200/\text{unit}$ ,  $C = \$20/\text{unit}$ ,  $C_1 = \$3/\text{unit}$ ,  $C_2 = \$10/\text{unit}/\text{week}$ ,  $C_3 = \$5/\text{unit}/\text{week}$ ,  $C_4 = \$25/\text{unit}$ ,  $\beta(t) = e^{-0.1t}$ ,  $a_1 = 100$ ,  $a_2 = 170$ ,  $b_1 = 5$ ,  $b_2 = 5$ ,  $D_0 = 130$ ,  $\mu_1 = 6$  weeks,  $\mu_2 = 8$  weeks,  $T = 12$  weeks,  $\theta(t) = 0.05t$ ,  $a = 200$ ,  $b = 1.5$ . Set  $p_{min} = 85$  and  $p_{max} = 95$ , then  $G(p_{max}) = -13296.6366 < 0$ ,  $G(p_{min}) = 20432.6806 > 0$ , using Newton-Raphson method, we have  $t_1^* = 4.2097$  weeks and  $p^* = \$91.0764$ . Then  $(t_1^*, p^*) \in D_1$ , thus  $ATP(t_1^*, p^*) = ATP_1(t_1^*, p^*) = \$256225.28$ ,  $S^* = 34741.64$  units, and  $Q^* = 78016.73$  units.

### 5.2 Example 2

This example is based on the following data:  $A_0 = \$200/\text{unit}$ ,  $C = \$20/\text{unit}$ ,  $C_1 = \$3/\text{unit}$ ,  $C_2 = \$10/\text{unit}/\text{week}$ ,  $C_3 = \$5/\text{unit}/\text{week}$ ,  $C_4 = \$25/\text{unit}$ ,  $\beta(t) = e^{-0.1t}$ ,  $a_1 = 100$ ,  $a_2 = 170$ ,  $b_1 = 5$ ,  $b_2 = 5$ ,  $D_0 = 130$ ,  $\mu_1 = 6$  weeks,  $\mu_2 = 10$  weeks,  $T = 12$  weeks,  $\theta(t) = 0.05t$ ,  $a = 200$ ,  $b = 1.5$ .  $p^* = \$91.0764$ . Initialize  $p_{min} = 80$  and  $p_{max} = 85$ ,  $G(p_{max}) > 0$ ,  $p^* = p_{max} = \$84.7$ ,  $t_1^* = \hat{t}_1 = 4.1069$  weeks, so  $ATP^* = \$250520.65$ ,  $S^* = 38588.99$  units, and  $Q^* = 88807.34$  units.

### 5.3 Example 3

This example is based on the following data:  $A_0 = \$200/\text{unit}$ ,  $C = \$20/\text{unit}$ ,  $C_1 = \$3/\text{per}$ ,  $C_2 = \$10/\text{unit}/\text{week}$ ,  $C_3 = \$5/\text{unit}/\text{week}$ ,  $C_4 = \$25/\text{unit}$ ,  $\beta(t) = e^{-0.1t}$ ,  $a_1 = 100$ ,  $a_2 = 170$ ,  $b_1 = 5$ ,  $b_2 = 5$ ,  $D_0 = 130$ ,  $\mu_1 = 6$  weeks,  $\mu_2 = 8$  weeks,  $T = 12$  weeks,  $\theta(t) = 0.05t$ ,  $a = 200$ ,  $b = 1.5$ . Set  $p_{min} = 95$  and  $p_{max} = 105$ . Then  $G(p_{min}) < 0$ ,  $p^* = p_{min} = \$95$ ,  $t_1^* = \tilde{t}_1 = 4.2901$  weeks, so  $ATP^* = \$252373.39$ ,  $S^* = 31297.26$  units, and  $Q^* = 68973.47$  units.

## 6 Conclusion

In this paper, considering customer demand depends on both trapezoidal time and price, we investigate an inventory model for deteriorating items in a fixed order cycle. Shortages are allowed and partial backlogging rate is a decreasing function of the customers waiting time during the shortage period. The existence and uniqueness of the optimal solution for the model are discussed, and an easy-to-operate algorithm is provided to research the optimal price, shortage time point, the maximum inventory level, and initial ordering quantity. Numerical examples are used to verify the correctness of the theoretical results.

There still exist some limitations in our research, an inventory model with multiple inventory cycle for the variable holding cost will be interesting. Moreover, some epochal inventory features such as variable inventory cycle, stochastic demand setting, shopping experience, low carbon regulation, will also be incorporated in this research.

**Acknowledgements.** This study was supported by Research Program of Tianjin Municipal Education Commission (No. 2017KJ242).

## References

- Lin, K.: An extended inventory models with trapezoidal type demands. *Appl. Math. Comput.* **219**, 11414–11419 (2013)
- Cheng, M., Zhang, B., Wang, G.: Optimal policy for deteriorating items with trapezoidal type demand and partial backlogging. *Appl. Math. Model.* **35**, 3552–3560 (2011)
- Cheng, M., Wang, G.: A note on the inventory model for deteriorating items with trapezoidal type demand rate. *Comput. Ind. Eng.* **56**, 1296–1300 (2009)
- Micheal, K., Rochford, L., Wotruba, T.: How new product introductions affect sales management strategy: the impact of type of newness of the new product. *J. Prod. Innov. Manag.* **20**, 270–283 (2003)
- Glock, C., Grosse, H.: Decision support models for production ramp-up: a systematic literature review. *Int. J. Prod. Res.* **53**(21–22), 6637–6651 (2015)
- Shah, N., Shah, D., Patel, D.: Optimal transfer, ordering and payment policies for joint supplier-buyer inventory model with price-sensitive trapezoidal demand and net credit. *Int. J. Syst. Sci.* **46**(9–12), 1752–1761 (2015)
- Singh, N., Vaish, B., Singh, S.: An EOQ model with pareto distribution for deterioration, trapezoidal type demand and backlogging under trade credit policy. *IUP J. Comput. Math.* **3**(4), 30–53 (2010)
- Uthayakumar, R., Rameswari, M.: An economic production quantity model for defective items with trapezoidal type demand rate. *J. Optim. Theor. Appl.* **154**, 1055–1079 (2012)
- Khan, M.A., Shaikh, A.A., Panda, G.C., Taleizadehe, A.A.: Inventory system with expiration date: pricing and replenishment decisions. *Comput. Ind. Eng.* **132**, 232–247 (2019)
- Maihami, R., Kamalabadi, I.N.: Joint pricing and inventory control for non-instantaneous deteriorating items with partial backlogging and time and price dependent demand. *Int. J. Prod. Econ.* **136**, 116–122 (2012)

- Dye, C.Y., Chang, H.J., Teng, J.T.: A deteriorating inventory model with time-varying demand and shortage-dependent partial backlogging. *Eur. J. Oper. Res.* **172**, 417–429 (2006)
- Teng, J., Chern, M., Yang, H.: An optimal recursive method for various inventory replenishment models with increasing demand and shortages. *Nav. Res. Logist.* **44**, 791–806 (1997)
- Yang, H., Teng, J., Chern, M.: Deterministic inventory lot-size models under inflation with shortages and deterioration for fluctuating demand. *Nav. Res. Logist.* **44**(2), 144–158 (2001)
- Abad, P.L.: Optimal price and order size under partial backordering incorporating shortage, backorder and lost sale costs. *Int. J. Prod. Econ.* **114**, 179–186 (2008)
- Abad, P.L.: Optimal pricing and lot sizing under conditions of perishability and partial backlogging. *Manage. Sci.* **42**, 1093–1104 (1996)
- Dye, C.: Joint pricing and ordering policy for a deteriorating inventory with partial backlogging. *Omega* **35**(2), 184–189 (2007)
- Zhou, Y., Lau, H., Yang, S.: A new variable production scheduling strategy for deteriorating items with time-varying demand and partial lost sale. *Comput. Oper. Res.* **30**, 1753–1776 (2003)
- Ghosh, S.K., Khanra, S., Chaudhuri, K.S.: Optimal price and lot size determination for a perishable product under conditions of finite production, partial backordering and lost sale. *Appl. Math. Comput.* **217**, 6047–6053 (2011)
- Khan, M.A., Shaikh, A.A., Panda, G.C., Taleizadeh, A.A.: Inventory system with expiration date: pricing and replenishment decisions. *Comput. Ind. Eng.* **132**, 232–247 (2019)
- Lashgari, M., Taleizadeh, A.A., Sadjadi, S.: Ordering policies for non-instantaneous deteriorating items under hybrid partial prepayment, partial delay payment and partial backordering. *J. Oper. Res. Soc.* **69**, 1167–1196 (2018)
- Salehi, H., Taleizadeh, A.A., Tavakkoli-Moghaddam, R.: An EOQ model with random disruptions and partial backordering. *Int. J. Prod. Res.* **54**, 2600–2609 (2016)
- Taleizadeh, A.A.: A constrained integrated imperfect manufacturing-inventory system with preventive maintenance and partial backordering. *Ann. Oper. Res.* **261**, 303–337 (2018)
- Taleizadeh, A.A., Khanbaglo, M.P.S., Barrón, C., Eduardo, L.: An EOQ inventory model with partial backordering and reparation of imperfect products. *Int. J. Prod. Econ.* **182**, 418–434 (2016)
- Xu, Y., Bisi, A., Dada, M.: A finite-horizon inventory system with partial backorders and inventory holdback. *Oper. Res. Lett.* **45**(4), 315–322 (2017)
- Xu, C., Zhao, D., Min, J., Hao, J.: An inventory model for nonperishable items with warehouse mode selection and partial backlogging under trapezoidal type demand. *J. Oper. Res. Soc.* **72**(4), 744–763 (2021)



# A Set-Theoretic Representation of Algebraic L-domains

Juan Zou, Yuhan Zhao, Cuixia Miao, and Longchun Wang<sup>(✉)</sup>

School of Mathematical Sciences, Qufu Normal University, Jining, China  
longchunw@163.com

**Abstract.** In this paper, we aim to establish a concrete representation, as a family of sets, for every algebraic L-domain. We generalize the notion of a topped algebraic intersection structure to a locally algebraic intersection structure. Just as topped algebraic intersection structures are concrete representations of algebraic lattices, locally algebraic intersection structures are concrete representations of algebraic L-domains. This result extends the classic Stone's representation theorem for Boolean algebras to the case of algebraic L-domains. In addition, it will be seen that many well-known representations of algebraic L-domains can be analyzed with the framework of locally algebraic intersection structures.

**Keywords:** Stone's representation theorem · Domain theory · Algebraic L-domain · Topped algebraic intersection structure

## 1 Introduction

The development of re-framing algebraic structures and order structures within the theory of sets can be traced back to Stone's representation theorem for Boolean algebras [13] and Birkhoff's representation theorem for finite distributive lattices [2]. Their results show that every Boolean algebra or finite distributive lattice can be represented as a family of sets. So far, many scholars have pointed out that various structures such as groups, rings, lattices and semilattices can be understood much better via the theory of sets.

Algebraic L-domains introduced by A. Jung [9], as good candidates for denotational semantics of programming languages, have been widely applied in theoretical computer science, especially in Domain theory. The category of algebraic L-domains with Scott continuous function is cartesian closed, as same as that of algebraic lattices with Scott continuous functions. Algebraic lattices are a proper subclass of algebraic L-domains, which have an elementary set-theoretic representation as topped algebraic intersection structures. An algebraic intersection structure  $\mathcal{L}$  on a set  $X$  is a non-empty family of subsets of  $X$  which satisfies (a):  $\bigcap_{i \in I} A_i \in \mathcal{L}$  for every non-empty family  $\{A_i\}_{i \in I}$  in  $\mathcal{L}$ , and (b):  $\bigcup_{i \in I} A_i \in \mathcal{L}$  for

---

This work is supposed by Shandong Provincial Natural Science Foundation(ZR2022MA022).

every directed family  $\{A_i\}_{i \in I}$  in  $\mathcal{L}$ . If  $\mathcal{L}$  also satisfies (c):  $X \in \mathcal{L}$ , then it is called a topped algebraic intersection structure. Moreover, algebraic intersection structures can be used to characterize Scott domains, another important subclass of algebraic L-domains.

The main purpose of this paper is to provide a set-theoretic representation of algebraic L-domains. An algebraic domain  $L$  is called an algebraic L-domain if it satisfies the local property that for every  $x \in L$ , the principal ideal  $\downarrow x$  is an algebraic lattice. Motivated by this observation and the set-theoretic representation of algebraic lattices, we define a locally algebraic intersection structure for every algebraic L-domain. The notion of a locally algebraic intersection structure generalizes that of algebraic intersection structure by simply changing Condition (a), within which the local property of an algebraic L-domain can be easily characterized. In Sect. 3, we show that every algebraic L-domain can be rewritten as a locally algebraic intersection structure. This enables us to perform algebraic L-domain in a pure set-theoretic form.

We realize that there are at least four different representations for algebraic L-domains, ranging from Chen and Jung’s disjunctive propositional logics [4], over Wu et al’s algebraic L-information systems [16] and algebraic L-closure spaces [17], to Guo et al’s LCF-contexts [8]. In Sect. 4, we give a brief review about these representations. Although we illustrate that an algebraic L-domain consisting of a family of sets with set inclusion may not be a locally algebraic intersection structure, all of the above four representations of algebraic domains are proven to be locally algebraic intersection structures. So these known representations for algebraic L-domains have a unified set-theoretic form.

## 2 Preliminary

We first recall some order and domain theoretical terminology that will be used in this paper, most of them come from [5, 7].

Let  $(P, \leq)$  be a poset and  $A \subseteq P$ . We denote by  $\downarrow A$  the down set  $\{x \in P \mid (\exists a \in A)x \leq a\}$ . If  $A$  is a singleton  $\{x\}$ , then we just write  $\downarrow x$ . The supremum of  $A$ , if it exists, is the least element of the set of all upper bounds of  $A$  in  $P$ . We denoted it by  $\bigvee A$ . The infimum  $\bigwedge A$  of  $A$  is defined dually. We denote  $x \wedge y$  in place of  $\bigwedge\{x, y\}$  when it exists. The poset  $P$  is said to be a *complete lattice* if every subset of it has an infimum. If  $x \wedge y$  exists for all  $x, y \in P$ , then  $P$  is called a *semilattice*. A non-empty subset  $D$  of  $P$  is said to be *directed* if for every  $x, y \in D$ , there is some  $z \in D$  such that  $x \leq z$  and  $y \leq z$ .

A poset is said to be a *dcpo* if every directed subset of it has a supremum. Let  $P$  be a dcpo. An element  $k \in P$  is said to be *compact*, if whenever  $D$  is directed with  $k \leq \bigvee D$ , then  $k \leq d$  for some  $d \in D$ . We denote by  $\mathcal{K}(P)$  the set of all compact elements of  $P$ . If every element in  $P$  is a directed supremum of compact elements, then  $P$  is called an *algebraic domain*.

**Definition 1.** ([9]) *An algebraic domain  $D$  is said to be an algebraic L-domain if for every element  $x$  of  $D$ , the principal ideal  $\downarrow x = \{y \in D \mid y \leq x\}$  is a complete lattice.*

### 3 Representation Theorem of L-domains

In this section, we give a concrete set-theoretic representation for every algebraic L-domain.

**Definition 2.** A non-empty family  $\mathcal{C}$  of subsets of a set  $X$  is said to be a locally algebraic intersection structure if

- (L1) for every directed family  $\{A_i \in \mathcal{C} \mid i \in I\}$ , the directed union  $\bigcup_{i \in I} A_i \in \mathcal{C}$ ,
- (L2) for every  $C \in \mathcal{C}$  and non-empty family  $\{A_j \in \mathcal{C} \mid A_j \subseteq C, j \in J\}$ , the intersection  $\bigcap_{j \in J} A_j \in \mathcal{C}$ .

Clearly, every algebraic intersection structure, especially every topped algebraic intersection structure, is a locally algebraic intersection structure.

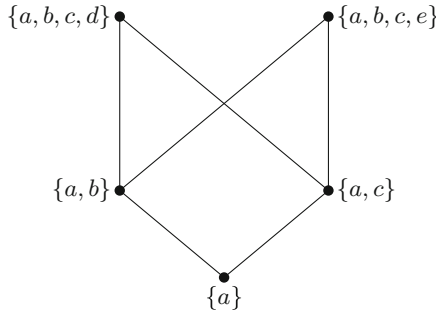


Fig. 1. The poset  $(\mathcal{C}, \subseteq)$  in Example 1

*Example 1.* Let  $X = \{a, b, c, d, e\}$  and  $\mathcal{C} = \{\{a\}, \{a, b\}, \{a, c\}, \{a, b, c, d\}, \{a, b, c, e\}\}$ . Then  $\mathcal{C}$  is a locally algebraic intersection structure on the set  $X$ . However,  $\mathcal{C}$  is not an algebraic intersection structure. In addition, the poset  $(\mathcal{C}, \subseteq)$  is defined as in Fig. 1.

*Remark 1.* Let  $\mathcal{C}$  be a locally algebraic intersection structure on a set  $X$ .

- (1) By Condition (L1),  $\mathcal{C}$  ordered by set inclusion forms a dcpo, in which the supremum of every directed family is given by set union.
- (2) For every  $B \in \mathcal{C}$  and every subset  $A$  of  $B$ , the family  $\{C \in \mathcal{C} \mid A \subseteq C \subseteq B\}$  is non-empty. Define

$$\Gamma_B(A) = \bigcap \{C \in \mathcal{C} \mid A \subseteq C \subseteq B\}. \tag{3.1}$$

Then  $\Gamma_B(A) \in \mathcal{C}$ , by Condition (L2). Moreover, it is easy to see that  $\Gamma_B(A) = A$  whenever  $A \in \mathcal{C}$ .



**Proposition 1.** *Let  $\mathcal{C}$  be a locally algebraic intersection structure on a set  $X$  and  $M$  a finite subset of  $X$ .*

- (1) *If  $B \in \mathcal{C}$  and  $M \subseteq B$ , then  $M \subseteq \Gamma_B(M) \subseteq B$ .*
- (2) *If  $B_1, B_2 \in \mathcal{C}$  and  $M \subseteq B_1 \subseteq B_2$ , then  $\Gamma_{B_1}(M) = \Gamma_{B_2}(M)$ .*
- (3) *If  $B, B_1, B_2 \in \mathcal{C}$ ,  $M \subseteq B \subseteq B_1$  and  $M \subseteq B \subseteq B_2$ , then  $\Gamma_{B_1}(M) = \Gamma_{B_2}(M)$ .*

*Proof.* (1) It is straightforward by Eq. (3.1).

(2) Suppose that  $B_1, B_2 \in \mathcal{C}$  and  $M \subseteq B_1 \subseteq B_2$ . Then it is easy to see that

$$\bigcap \{C \in \mathcal{C} \mid M \subseteq C \subseteq B_1\} = \bigcap \{C \in \mathcal{C} \mid M \subseteq C \subseteq B_2\}.$$

Thus by Eq. (3.1), we have  $\Gamma_{B_1}(M) = \Gamma_{B_2}(M)$ .

(3) Suppose that  $B, B_1, B_2 \in \mathcal{C}$ . If  $M \subseteq B \subseteq B_1$  and  $M \subseteq B \subseteq B_2$ , then  $\Gamma_B(M) = \Gamma_{B_1}(M)$  and  $\Gamma_B(M) = \Gamma_{B_2}(M)$ , using part (2) twice. Therefore,  $\Gamma_{B_1}(M) = \Gamma_{B_2}(M)$ .

**Lemma 1.** *Let  $\mathcal{C}$  be a locally algebraic intersection structure on a set  $X$ .*

(1) *For every  $C \in \mathcal{C}$ , the family*

$$\mathcal{D} = \{\Gamma_C(M) \mid M \text{ is a finite subset of } C\} \tag{3.2}$$

*is directed under set inclusion and  $C = \bigcup \mathcal{D}$ .*

(2) *For the dcpo  $(\mathcal{C}, \subseteq)$ ,*

$$\mathcal{K}(\mathcal{C}) = \{\Gamma_C(M) \mid C \in \mathcal{C} \text{ and } M \text{ is a finite subset of } C\}. \tag{3.3}$$

*Proof.* (1) The family  $\mathcal{D}$  is non-empty since  $\emptyset$  is a finite set of  $C$  and  $\Gamma_C(\emptyset) \in \mathcal{D}$ . Let  $\Gamma_C(M_1), \Gamma_C(M_2) \in \mathcal{D}$ , where  $M_1, M_2$  are finite subsets of  $C$ . Then  $M_1 \cup M_2$  is a finite subset of  $C$  and  $\Gamma_C(M_1 \cup M_2) \in \mathcal{D}$ . By Eq. (3.1), it is clear that  $\Gamma_C(M_1) \subseteq \Gamma_C(M_1 \cup M_2)$  and  $\Gamma_C(M_2) \subseteq \Gamma_C(M_1 \cup M_2)$ . This shows that the family  $\mathcal{D}$  is directed.

By part (1) of Proposition 1,  $\Gamma_C(M) \subseteq C$  for every finite subset  $M \subseteq C$ . Thus,  $\bigcup \mathcal{D} \subseteq C$ . Conversely, for every  $x \in C$ , we have

$$x \in \{x\} \subseteq \Gamma_C(\{x\}) \subseteq \bigcup \mathcal{D},$$

which implies that  $C \subseteq \bigcup \mathcal{D}$ .

(2) Suppose that  $C \in \mathcal{C}$  and  $M$  is a finite subset of  $C$ . If  $\Gamma_C(M) \subseteq \bigcup \mathcal{D}$  for some directed subfamily of  $\mathcal{C}$ , then  $M \subseteq \Gamma_C(M) \subseteq C$  and  $M \subseteq \Gamma_C(M) \subseteq \bigcup \mathcal{D}$ . Thus  $\Gamma_{\bigcup \mathcal{D}}(M) = \Gamma_C(M)$ , by part (3) of Proposition 1. Because  $\mathcal{D}$  is directed and  $M$  is a finite subset of  $\bigcup \mathcal{D}$ , there is some  $D_0 \in \mathcal{D}$  such that  $M \subseteq D_0$ . Hence  $\Gamma_{\bigcup \mathcal{D}}(M) \subseteq \Gamma_{\bigcup \mathcal{D}}(D_0)$ . But  $\Gamma_{\bigcup \mathcal{D}}(D_0) = D_0$ , since  $D_0 \in \mathcal{C}$ . So  $\Gamma_{\bigcup \mathcal{D}}(M) \subseteq D_0$ , which indicates that  $\Gamma_C(M)$  is a compact element in  $(\mathcal{C}, \subseteq)$ .

Conversely, suppose that  $C$  is a compact element in  $(\mathcal{C}, \subseteq)$ . By part (1), we have

$$C = \bigcup \{\Gamma_C(M) \mid M \text{ is a finite subset of } C\}$$

and the family  $\{\Gamma_C(M) \mid M \text{ is a finite subset of } C\}$  is directed. Invoke the compactness of  $C$  in  $\mathcal{C}$  to find a finite set  $M_0 \subseteq C$  such that  $C \subseteq \Gamma_C(M_0)$ . Thus,  $C = \Gamma_C(M_0)$ , since the reverse inclusion holds from part (1) of Proposition 1.

**Theorem 1.** *Let  $\mathcal{C}$  be a locally algebraic intersection structure on a set  $X$ . Then the dcpo  $(\mathcal{C}, \subseteq)$  is an algebraic L-domain.*

*Proof.* By Lemma 1, the dcpo  $(\mathcal{C}, \subseteq)$  is an algebraic domain in which the compact elements are of the form  $\Gamma_C(M)$ , where  $C \in \mathcal{C}$  and  $M$  is a finite subset of  $\mathcal{C}$ . So it suffices to show that the family

$$\downarrow B = \{C \in \mathcal{C} \mid C \subseteq B\}$$

with set inclusion is a complete lattice for every  $B \in \mathcal{C}$ .

Condition (L2) implies that the family  $\downarrow B$  is closed under non-empty intersections, and  $B \in \downarrow B$  guarantees that  $\downarrow B$  is closed under the empty intersection. Then  $\downarrow B$  is a complete lattice in which the infimum of every subfamily is given by set intersection.

The converse of Theorem 1 does not hold.

*Example 2.* Let  $X = \mathbb{N} \cup \{-1\}$ , where  $\mathbb{N}$  is the set of natural numbers. Set

$$A_0 = \{0\}, A_1 = \{0, 1\}, A_2 = \{0, 1, 2\}, \dots, A_n = \{0, 1, 2, \dots, n\}, \dots$$

and set

$$\mathcal{C} = \{A_i \mid i \in \mathbb{N}\} \cup \{X\}.$$

Then it is readily to see that  $(\mathcal{C}, \subseteq)$  is an algebraic L-domain. However,  $\mathcal{C}$  is not a locally algebraic intersection structure, since the union of the directed family  $\{A_0, A_1, A_2, \dots, A_n, \dots\}$  is equal to  $\mathbb{N}$ , which is not an element of  $\mathcal{C}$ .

**Theorem 2 (Representation Theorem).** *Let  $(L, \leq)$  be an algebraic L-domain. Then there is a locally algebraic intersection structure that is order-isomorphic to  $L$ .*

*Proof.* For every  $a \in L$ , define

$$D_a = \{k \in \mathcal{K}(L) \mid k \leq a\}. \tag{3.4}$$

We first show that the family

$$\mathcal{C}_L = \{D_a \mid a \in L\} \tag{3.5}$$

ordered by set inclusion is order-isomorphic to  $(L, \leq)$ .

Define a function  $f : L \rightarrow \mathcal{C}_L$  by

$$f(a) = D_a. \tag{3.6}$$

Obviously,  $f$  is well-defined and surjective. Suppose that  $D_a \subseteq D_b$ , where  $a, b \in L$ . Because  $(L, \leq)$  is an algebraic L-domain, we have  $a = \bigvee D_a$  for every  $a \in L$ . Thus

$$a = \bigvee D_a \leq \bigvee D_b = b.$$

Conversely, suppose that  $a \leq b \in L$ . Then it is clear that  $D_a \subseteq D_b$  by Eq. (3.4). Therefore, the function  $f : L \rightarrow \mathcal{C}_L$  is an order-isomorphism.

We next show that  $\mathcal{C}_L$  is a locally algebraic intersection structure.

Let  $\{D_{a_i} \in \mathcal{C}_L \mid i \in I\}$  be a directed subfamily of  $\mathcal{C}_L$ . As we have seen that the function  $f : L \rightarrow \mathcal{C}_L$  defined by Eq. (3.6) is an order-isomorphism, the indexing set  $\{a_i \in L \mid i \in I\}$  is a directed subset of  $L$ . Take  $a = \bigvee_{i \in I} a_i$ . Then we have

$$\begin{aligned} k \in D_a &\Leftrightarrow k \in \mathcal{K}(L) \text{ and } k \leq a = \bigvee_{i \in I} a_i \\ &\Leftrightarrow k \leq a_i \text{ for some } i \in I \\ &\Leftrightarrow k \in \bigcup_{i \in I} D_{a_i}. \end{aligned}$$

This implies that  $\bigcup_{i \in I} D_{a_i}$  is equal to  $D_a$  and hence is an element of  $\mathcal{C}_L$ .

Let  $D_a \in \mathcal{C}_L$ , where  $a \in L$ . Suppose that  $\{D_{a_j} \in \mathcal{C}_L \mid j \in J\}$  is a subfamily of  $\mathcal{C}_L$  such that  $D_{a_j} \subseteq D_a$  for every  $j \in J$ . Then  $\{a_j \mid j \in J\}$  is a subset of  $L$  and  $a_j \leq a$  for every  $j \in J$ . Since  $(L, \leq)$  is an algebraic L-domain, the set  $\downarrow a = \{x \in L \mid x \leq a\}$  is a complete lattice in the induced ordering. This implies that the set  $\{a_j \mid j \in J\}$  has an infimum in  $\downarrow a$ , say  $a_0$ . We claim that  $\bigcap_{j \in J} D_{a_j} = D_{a_0}$ . Indeed,

$$\begin{aligned} k \in \bigcap_{j \in J} D_{a_j} &\Leftrightarrow k \in \mathcal{K}(L) \text{ and } k \in D_{a_j} \text{ for all } j \in J \\ &\Leftrightarrow k \in \mathcal{K}(L) \text{ and } k \leq a_j \text{ for all } j \in J \\ &\Leftrightarrow k \in \mathcal{K}(L) \text{ and } k \leq a_0 \\ &\Leftrightarrow k \in D_{a_0}. \end{aligned}$$

So the intersection  $\bigcap_{j \in J} D_{a_j}$  belongs to  $\mathcal{C}_L$ .

## 4 Further Representations

In this section, we give an overview of representations of algebraic L-domains, relating locally algebraic intersection structures with some well-known formalisms from logic, information systems, closure spaces and Formal Concept Analysis.

### 4.1 Logical Algebras

The most conspicuous of characterizing domains as logical theory includes the work of such scholars as Abramsky, Zhang, Chen and Jung [1, 4, 18]. In [4], Chen and Jung built a framework of disjunctive propositional logic and showed how to use its Lindenbaum algebras to represent algebraic L-domains.

**Definition 3.** ([4]) *Let  $(L, \wedge, 0_L, 1_L)$  be a semilattice with least element  $0_L$  and greatest element  $1_L$ .*

- (1)  $x, y \in L$  are said to be disjoint if  $x \wedge y = 0_L$ .
- (2) A subset  $A$  of  $L$  is said to be disjoint if  $x \wedge y = 0_L$  for every distinct elements  $x, y$  in  $A$ .
- (3) The semilattice is said to be a  $D$ -semilattice if every disjoint subset  $A \subseteq L$  has a supremum  $\dot{\bigvee} A$ , where, we use  $\dot{\bigvee} A$  to denote the supremum of a disjoint set  $A$ .
- (4) The semilattice is called a  $dD$ -semilattice if it is a  $D$ -semilattice such that

$$x \wedge (\dot{\bigvee} A) = \dot{\bigvee}_{a \in A} (x \wedge a)$$

for each element  $x \in L$  and disjoint subset  $A$  of  $L$ .

**Definition 4.** ([4]) Let  $L$  be a  $dD$ -semilattice.

- (1) An element  $a \in L$  is called coprime if, for every disjoint subset  $A$  of  $L$ ,  $a \leq \dot{\bigvee} A$  implies  $a \leq x$  for some  $x \in A$ . The set of coprime elements is denoted by  $\text{Cp}(L)$ .
- (2) The  $dD$ -semilattice is said to be coprime generated if for each  $x \in L$ , there is a unique disjoint subset  $A \subseteq \text{Cp}(L)$  such that  $x = \dot{\bigvee} A$ .

**Definition 5.** ([4]) Let  $L$  be a coprime generated  $dD$ -semilattice. A disjunctive completely prime filter  $F$  is a proper subset of  $L$  that satisfies the following conditions:

- (pt1) if  $x \in F$  and  $x \leq y \in L$ , then  $y \in F$ ;
- (pt2) if  $x, y \in F$ , then  $x \wedge y \in F$ ;
- (pt3) if  $\dot{\bigvee} A \in F$  for some disjoint subset  $A$  of  $L$ , then there is some  $a \in A$  such that  $a \in F$ .

The family of disjunctive completely prime filters is denoted by  $\text{pt}(L)$ .

**Theorem 3.** ([3,4]) If  $L$  is a coprime generated  $dD$ -semilattice, then  $\text{pt}(L)$  is an algebraic  $L$ -domain. Moreover, every algebraic  $L$ -domain can be generated in this way up to isomorphism.

In [4], Chen and Jung built a logical system which is logical complete with respect to  $dD$ -semilattices. Theorem 3 therefore provides a logical characterization for algebraic  $L$ -domains. In their programme, the family of disjunctive completely prime filters for a coprime generated  $dD$ -semilattice plays a central role. Now we show that the family of disjunctive completely prime filters is a locally algebraic intersection structure.

**Theorem 4.** Let  $L$  be a coprime generated  $dD$ -semilattice. Then  $\text{pt}(L)$  is a locally algebraic intersection structure.

*Proof.* Let  $\{F_i \mid i \in I\}$  be a directed family of  $\text{pt}(L)$ . We show that the union  $\bigcup_{i \in I} F_i$  is also an element of  $\text{pt}(L)$  by checking the conditions for a disjunctive completely prime filter. We illustrate this for Condition  $\text{pt}(3)$ , since the others

are similar. Suppose that  $\dot{\bigvee} A \in \bigcup_{i \in I} F_i$  for some disjoint subset  $A$  of  $L$ . Then there exists some  $i_0 \in I$  such that  $\dot{\bigvee} A \in F_{i_0}$ . Using Condition pt(3) for the disjunctive completely prime filter  $F_{i_0}$ , it follows that  $a \in F_{i_0} \subseteq \bigcup_{i \in I} F_i$  for some  $a \in A$ .

Let  $\{F_j \mid j \in J\}$  be a non-empty subfamily of  $\text{pt}(L)$  in which every element is contained in another disjunctive completely prime filter  $G$ . Set

$$F = \bigcap \{F_j \mid j \in J\}.$$

It is clear that  $F$  satisfies Conditions pt(1) and pt(2). For Condition pt(3), suppose that  $\dot{\bigvee} A \in F$  for some disjoint subset  $A$  of  $L$ . Then  $\dot{\bigvee} A \in F_j$  for every  $j \in J$ , and hence  $\dot{\bigvee} A \in G$ . This implies that there exists some  $a \in A$  such that  $a \in G$ . We claim that  $a \in F_j$  for all  $j \in J$ . Indeed, fixing  $j \in J$ ,  $\dot{\bigvee} A \in F_j$  implies that there is some  $a_j \in A$  such that  $a_j \in F_j$ , since  $F_j$  is a disjunctive completely prime filter. If  $a_j \neq a$ , then  $a_j \wedge a = 0_L$ . Because both  $a$  and  $a_j$  in  $G$ , it follows by Condition pt(2) that  $0_L \in G$ . Thus  $G = L$ , a contradiction.

### 4.2 Information Systems

In [10], D. Scott introduced information systems as a concrete representation for Scott domains which turns out to be of remarkable significance for understanding the relationship between program logic and denotational semantics. Since then, many similar information systems have been presented to capture other domains [12, 14, 15].

**Definition 6.** [11] *Let  $A$  be a set,  $\text{Con}$  a family of finite subsets of  $A$  and  $\vdash$  a binary relation from  $\text{Con}$  to  $A$ . Then  $(A, \text{Con}, \vdash)$  is called an algebraic information system if the following conditions hold for every  $M, N \in \text{Con}$ :*

- (I1)  $a \in A \Rightarrow \{a\} \in \text{Con}$ ,
- (I2)  $M \vdash a \Rightarrow M \cup \{a\} \in \text{Con}$ ,
- (I3)  $N \subseteq M$  and  $N \vdash a \Rightarrow M \vdash a$ ,
- (I4)  $(M \vdash N, N \vdash a) \Rightarrow M \vdash a$ ,
- (I5)  $M \vdash N \Rightarrow (\exists M_1 \in \text{Con}) M \vdash M_1 \vdash M_1 \vdash N$ ,
- (I6)  $(\forall F \sqsubseteq A) M \vdash F \Rightarrow (\exists N \in \text{Con})(M \vdash N, F \subseteq N)$ ,

where  $F \sqsubseteq A$  means that  $F$  is a finite subset of  $A$  and  $M \vdash F$  means that  $M \vdash b$  for all  $b \in F$ .

In an algebraic information system  $(A, \text{Con}, \vdash)$ , we define

$$\overline{X} = \{a \in A \mid (\exists M \in \text{Con})(M \subseteq X, M \vdash a)\} \tag{4.1}$$

for every  $X \subseteq A$ .

**Definition 7.** [16] *Let  $(A, \text{Con}, \vdash)$  be an algebraic information system, a non-empty subset  $S \subseteq A$  is called a state if it satisfies the following conditions:*

- (S1)  $\overline{S} \subseteq S$ ,
- (S2)  $(\forall F \sqsubseteq S)(\exists M \in \text{Con})(M \subseteq S, M \vdash F)$ .

As usual, we use  $|\mathbf{A}|$  to denote the family of all states of an algebraic information system  $(A, \text{Con}, \vdash)$ .

**Definition 8.** [16] *An algebraic information system  $(A, \text{Con}, \vdash)$  is said to be an algebraic L-information system if, for every  $M \in \text{Con}$  and  $F \sqsubseteq \overline{M}$ , there is  $N \in \text{Con}$  such that*

- (IL1)  $\overline{F} \subseteq \overline{N}$  and  $N \subseteq \overline{M}$
- (IL2) for every  $M_1 \in \text{Con}$ ,  $\overline{F} \subseteq \overline{M_1} \subseteq \overline{M}$  can always implies that  $\overline{N} \subseteq \overline{M_1}$ .

We call  $N$  an  $M$ -sup of  $F$  and denote the set of all  $M$ -sup of  $F$  by  $\Sigma(M, F)$ .

**Lemma 2.** [16] *Let  $S$  be a state of an algebraic L-information system  $(A, \text{Con}, \vdash)$ ,  $M_1, M_2 \in \text{Con}$  and  $F \sqsubseteq A$ . If  $M_1, M_2 \subseteq S$  and  $F \subseteq \overline{M_1} \cap \overline{M_2}$ , then  $\overline{N_1} = \overline{N_2}$  for all  $N_1 \in \Sigma(M_1, F)$  and  $N_2 \in \Sigma(M_2, F)$ .*

In [11, Proposition 32], Spren et al. established a representation of algebraic domains in terms of algebraic information systems; and in [16, Theorems 3.1 and 3.3], Wu et al. provided a representation of continuous L-domains. As a direct consequence of these results, we have:

**Corollary 1.** *If  $(A, \text{Con}, \vdash)$  is an algebraic L-information system, then  $|\mathbf{A}|$  ordered by set inclusion forms an algebraic L-domain. Moreover, every algebraic L-domain can be generated in this way up to isomorphism.*

The following theorem tells us that this representation essentially defines a locally algebraic intersection structure.

**Theorem 5.** *Let  $(A, \text{Con}, \vdash)$  be an algebraic L-information system. Then  $|\mathbf{A}|$  is a locally algebraic intersection structure.*

*Proof.* Suppose that the family  $\{S_i \in |\mathbf{A}| \mid i \in I\}$  is directed. For every  $a \in \bigcup_{i \in I} \overline{S_i}$ , by Equation (4.1), there is  $M \in \text{Con}$  such that  $M \sqsubseteq \bigcup_{i \in I} S_i$  and  $M \vdash a$ . Since  $\{S_i \in |\mathbf{A}| \mid i \in I\}$  is directed,  $M \sqsubseteq S_{i_0}$  for some  $i_0 \in I$ . Thus  $a \in \overline{S_{i_0}} \subseteq S_{i_0} \subseteq \bigcup_{i \in I} S_i$ . This implies that  $\bigcup_{i \in I} S_i$  satisfies Condition (S1). To show that  $\bigcup_{i \in I} S_i$  belongs to  $|\mathbf{A}|$ , it suffices to check that  $\bigcup_{i \in I} S_i$  also satisfies Condition (S2). For every  $F \sqsubseteq \bigcup_{i \in I} S_i$ , there is some  $i_1 \in I$  such that  $F \sqsubseteq S_{i_1}$ . Using Condition (S2) for the state  $S_{i_1}$ , it follows that  $M \subseteq S_{i_1}$  and  $M \vdash F$  for some  $M \in \text{Con}$ . We thus find  $M \in \text{Con}$  that satisfies  $M \subseteq \bigcup_{i \in I} S_i$  and  $M \vdash F$ . Condition (S2) follows.

Suppose that  $S \in |\mathbf{A}|$  and the family  $\{S_j \in |\mathbf{A}| \mid S_j \subseteq S, j \in J\}$  is non-empty. We show that  $\bigcap_{j \in J} S_j \in |\mathbf{A}|$  by checking that  $\bigcap_{j \in J} S_j$  is non-empty and satisfies Conditions (S1) and (S2).

Note that  $S_j$  is non-empty for every  $j \in J$ . Take  $a_j \in S_j$ . Then  $\{a_j\} \in \text{Con}$  and  $\emptyset \sqsubseteq \overline{\{a_j\}} \subseteq \overline{S_j} \subseteq S_j$ . By Definition 8, there is some  $N \in \Sigma(\{a_j\}, \emptyset)$  such

that  $N \subseteq \overline{\{a_j\}}$ . Thus  $\overline{N} \subseteq \overline{\{a_j\}} \subseteq \overline{S_j} \subseteq S_j$ . This implies that  $\overline{N} \subseteq \bigcap_{j \in J} S_j$ , by Lemma 2. So  $\bigcap_{j \in J} S_j$  is non-empty.

If  $a \in \bigcap_{j \in J} \overline{S_j}$ , then there is some  $M \in \text{Con}$  such that  $M \vdash a$  and  $M \subseteq \bigcap_{j \in J} S_j \subseteq S_j$  for every  $j \in J$ . Thus  $a \in \overline{S_j} \subseteq S_j$  for every  $j \in J$ , and hence  $a \in \bigcap_{j \in J} S_j$ . Condition (S1) follows. For Condition (S2), let  $F \sqsubseteq \bigcap_{j \in J} S_j$ . Then  $F \sqsubseteq S_j$  for every  $j \in J$ . Using Condition (S2) for  $F \sqsubseteq S_j$ , there is some  $M_j \in \text{Con}$  such that  $M_j \subseteq S_j$  and  $M_j \vdash F$  for every  $j \in J$ . By Lemma 2,  $\overline{N_j}$  are coincide for all  $N_j \in \Sigma(M_j, F)$  and all  $j \in J$ . Note that  $\overline{N_j} \subseteq \overline{S_j} \subseteq S_j$ . It follows that  $\overline{N_j} \subseteq \bigcap_{j \in J} S_j$ . Since  $N_j \vdash F$ , there are  $M_1, N_1 \in \text{Con}$  such that  $N_j \vdash M_1 \vdash N_1$  and  $F \subseteq N_1$  by Conditions (I6) and (I5). Therefore,  $M_1 \in \text{Con}$ ,  $M_1 \subseteq \bigcap_{j \in J} S_j$  and  $M_1 \vdash F$ .

### 4.3 Closure Spaces

Closure space is a useful interdisciplinary tool to restructure lattices. A classical result is that closure spaces generate exactly all of complete lattices, which becomes an inspiring source for many mathematicians. The idea of representing other order structures by a closure space would be traced back to Birkhoff’s representation theorem for finite distributive lattices [2]. Recently, Wu et al. [17] generalized the notion of a closure space to an algebraic L-closure space and developed the representation theory of finite distributive lattices to that of algebraic L-domains.

**Definition 9.** ([5]) Let  $X$  be a set. A closure operator on  $X$  is a function  $\gamma$  on  $\mathcal{P}(X)$  that satisfies: for every  $A, B \subseteq X$ ,

- (1)  $A \subseteq \gamma(A)$ ,
- (2)  $A \subseteq B \Rightarrow \gamma(A) \subseteq \gamma(B)$ ,
- (3)  $\gamma(A) = \gamma(\gamma(A))$ .

The set of all fixed-points of  $\gamma$  is denoted as  $\mathfrak{X}_\gamma$  and the pair  $(X, \mathfrak{X}_\gamma)$  is called a closure space.

**Definition 10.** ([5]) A closure space  $(X, \mathfrak{X}_\gamma)$  is said to be algebraic if for  $A \subseteq X$ ,

$$\gamma(A) = \bigcup \{ \gamma(F) \mid F \sqsubseteq A \}. \tag{4.2}$$

**Definition 11.** ([17]) Let  $(X, \mathfrak{X}_\gamma)$  be an algebraic closure space. An element  $C \in \mathfrak{X}_\gamma$  is said to be Finset-bounded if for every  $F \sqsubseteq C$ , there is  $a \in C$  such that  $F \subseteq \gamma(a) \subseteq C$ .

We use  $\mathcal{S}(\mathfrak{X}_\gamma)$  to denote the family of all FinSet-bounded subsets of  $(X, \mathfrak{X}_\gamma)$ .

**Definition 12.** ([17]) An algebraic closure space  $(X, \mathfrak{X}_\gamma)$  is said to be an algebraic L-closure space if for every  $x \in X$  and  $F \subseteq \gamma(x)$ , there is  $y \in \gamma(x)$  such that

- (LC1)  $F \subseteq \gamma(y)$ ;
- (LC2)  $z \in \gamma(x)$  and  $F \subseteq \gamma(z)$  implies that  $\gamma(y) \subseteq \gamma(z)$ .

**Theorem 6.** (*[17]*) *If  $(X, \mathfrak{X}_\gamma)$  is an algebraic L-closure space, then  $\mathcal{S}(\mathfrak{X}_\gamma)$  ordered by set inclusion forms an algebraic L-domain. Moreover, every algebraic L-domain can be generated in this way up to isomorphism.*

The above theorem demonstrates the capability of closure spaces in representing algebraic L-domains. In fact, the family of all FinSet-bounded subsets of an algebraic L-closure space is a locally algebraic intersection structure.

**Theorem 7.** *Let  $(X, \mathfrak{X}_\gamma)$  be an algebraic L-closure space. Then  $\mathcal{S}(\mathfrak{X}_\gamma)$  is a locally algebraic intersection.*

*Proof.* The proof is similar to that of Theorem 5.

#### 4.4 Formal Concept Analysis

Formal Concept Analysis was introduced by R. Wille in the 1980s as a mathematical theory for the formalization of conceptual thinking [6]. A fundamental application of Formal Concept Analysis is to restructure lattice theory, which needs the notion of a formal context.

A *formal context* is a triple  $(P_o, P_a, \vDash_P)$ , in which  $\vDash_P$  is a binary relation from the set  $P_o$  to the set  $P_a$ . In this case, two operators can be defined as follows:

$$\alpha : \mathcal{P}(P_o) \rightarrow \mathcal{P}(P_a), A \mapsto \{n \in P_a \mid \forall m \in A, m \vDash n\}, \tag{4.3}$$

$$\omega : \mathcal{P}(P_a) \rightarrow \mathcal{P}(P_o), B \mapsto \{m \in P_o \mid \forall n \in B, m \vDash n\}. \tag{4.4}$$

**Definition 13.** (*[8]*) *Let  $(P_o, P_a, \vDash_P)$  be a formal context and  $\mathcal{F}$  a non-empty family of non-empty finite subset of  $P_o$ . Then  $(P_o, P_a, \vDash_P, \mathcal{F})$  is said to be a consistent F-augmented context if, for every  $X \in \mathcal{F}$ , there is a directed family of  $\{X_i \in \mathcal{F} \mid i \in I\}$  such that  $\omega \circ \alpha(X) = \bigcup_{i \in I} X_i$ .*

For every  $A \subseteq P_o$ , we denote by  $\langle A \rangle$  the set  $\bigcup \{\omega \circ \alpha(X) \mid X \in \mathcal{F}, X \sqsubseteq A\}$ .

**Definition 14.** (*[8]*) *A consistent F-augmented context  $(P_o, P_a, \vDash_P, \mathcal{F})$  is said to be an LCF-context if, for every  $X \in \mathcal{F}$  and  $F \sqsubseteq \omega \circ \alpha(X)$ , there is  $Z \in \mathcal{F}$  such that*

- (CF1)  $\langle F \rangle \subseteq \omega \circ \alpha(Z) \subseteq \omega \circ \alpha(X)$ ;
- (CF2)  $Y \in \mathcal{F}$  and  $\langle F \rangle \subseteq \omega \circ \alpha(Y) \subseteq \omega \circ \alpha(X)$  implies that  $\omega \circ \alpha(Z) \subseteq \omega \circ \alpha(Y)$ .

**Definition 15.** (*[8]*) *Let  $(P_o, P_a, \vDash_P, \mathcal{F})$  be an LCF-contexts. A subset  $E$  of  $P_o$  is said to be an F-approximable extent if  $E = \langle E \rangle$  and the set  $\{\langle F \rangle \mid F \in \mathcal{F}, F \sqsubseteq E\}$  is directed.*

We denote by  $\mathfrak{C}(P)$  the family of all F-approximable extents of  $(P_o, P_a, \vDash_P, \mathcal{F})$ .



**Theorem 8.** (*[8]*) *If  $(P_o, P_a, \models_P, \mathcal{F})$  is an LCF-context, then  $\mathfrak{C}(P)$  ordered by set inclusion forms an algebraic L-domain. Moreover, every algebraic L-domain can be generated in this way up to isomorphism.*

Theorem 8 restructures algebraic L-domains in terms of Formal Concept Analysis. This method can also be included into the framework of a locally algebraic intersection structure.

By a similar process of the proof for Theorem 5, we can show that:

**Theorem 9.** *Let  $(P_o, P_a, \models_P, \mathcal{F})$  be an LCF-context. Then  $\mathfrak{C}(P)$  is a locally algebraic intersection structure.*

## References

1. Abramsky, S.: Domain theory in logical form. *Ann. Pure Appl. Logic* **51**, 1–77 (1991)
2. Birkhoff, G.: Rings of sets. *Duke Math. J.* **3**(3), 443–454 (1937)
3. Chen, Y.: Stone duality and representation of stable domain. *Comput. Math. Appl.* **34**(1), 27–41 (1997)
4. Chen, Y., Jung, A.: A logical approach to stable Domains. *Theor. Comput. Sci.* **368**, 124–148 (2006)
5. Davey, B.A., Priestly, H.A.: *Introduction to Lattices and Order*. Cambridge University Press, Cambridge (2002)
6. Ganter, B., Wille, R.: *Formal Concept Analysis*. Springer, Heidelberg (1999). <https://doi.org/10.1007/978-3-642-59830-2>
7. Gierz, G., Hofmann, K.H., Keimel, K., Lawson, J.D., Mislove, M., Scott, D.S.: *Continuous Lattices and Domains*. Cambridge University Press, Cambridge (2003)
8. Guo, L., Li, Q., Yao, L.: Locally complete consistent F-augmented contexts: a category-theoretic representation of algebraic L-domains. *Discrete Appl. Math.* **249**, 53–63 (2018)
9. Jung, A.: *Cartesian closed categories of domains*. CWI Tracts, vol. 66. Centrum voor Wiskunde en Informatica, Amsterdam (1989)
10. Scott, D.S.: Domains for denotational semantics. *Lect. Notes Comput. Sci.* **140**, 577–613 (1982)
11. Spreen, D., Xu, L., Mao, X.: Information systems revisited: the general continuous case. *Theor. Comput. Sci.* **405**, 176–187 (2008)
12. Spreen, D.: Generalised information systems capture L-domains. *Theor. Comput. Sci.* **869**, 1–28 (2021)
13. Stone, M.H.: The theory of representations for Boolean algebras. *Trans. Am. Math. Soc.* **40**, 37–111 (1936)
14. Wang, L., Li, Q.: Representations of stably continuous semi-lattices by information systems and abstract bases. *Info. Process. Lett.* **165**, 106036 (2021)
15. Wang, L., Zhou, X., Li, Q.: Information systems for continuous semi-lattices. *Theor. Comput. Sci.* **913**, 138–150 (2022)
16. Wu, M., Guo, L., Li, Q.: A representation of L-domains by information systems. *Theor. Comput. Sci.* **612**, 126–136 (2016)
17. Wu, M., Guo, L., Li, Q.: New representations of algebraic domains and algebraic L-domains via closure systems. *Semigroup Forum* **103**, 700–712 (2021)
18. Zhang, G.-Q.: *Logic of Domains*. Birkhauser, Boston (1991)



# Normality, Randomness and Kolmogorov Complexity of Continued Fractions

Prateek Vishnoi<sup>(✉)</sup>

Department of Computer Science and Engineering, Indian Institute of Technology  
Kanpur, Kanpur 208016, Uttar Pradesh, India  
[pratvish@cse.iitk.ac.in](mailto:pratvish@cse.iitk.ac.in)

**Abstract.** Recently, it has been shown that binary expansion of a number in a unit interval is Martin-Löf random if and only if its continued fraction is Martin-Löf random. In contrast, we show that if the continued fraction expansion of a number in a unit interval is Martin-Löf random then it is continued fraction normal, however, the converse need not be true. We also study a notion of Kolmogorov complexity for continued fraction.

**Keywords:** Martin-löf random · Continued fraction normal · Kolmogorov complexity

## 1 Introduction

Kolmogorov complexity concerns itself with the definition of randomness of finite and infinite objects [7], usually strings and sequences drawn from some finite alphabet. Recently, Nandakumar and Vishnoi [9] have studied the randomness of individual continued fractions, establishing relations between Martin-Löf randomness, computable randomness and the continued fraction expansion. They show that these randomness notions are invariant with respect to whether the underlying real is represented using the base- $b$  expansion, for any  $b \geq 2$ , or the continued fraction expansion.

On the other hand, in certain restricted notions of randomness, these notions may not coincide, like that of normality. The notion of a normal sequence, first defined by Borel [3], requires that in base- $b$  sequence, every finite block  $w$  of digits occurs with asymptotic frequency  $b^{-|w|}$ . To generalize this notion to normal continued fractions, an appropriate probability to consider is the Gauss measure (see [4, 5]) - we require that every finite string of positive integers occur with asymptotic frequency equal to the Gauss measure of the cylinder determined by that string. It is known that the set of reals in  $[0, 1]$  which are continued fraction normal, has Gauss measure 1. It is known in [11, 13] that continued fraction normality and base- $b$  normality do not coincide.

It is shown in Nandakumar, Vishnoi [9] that every Martin-Löf random continued fraction and computably random continued fraction is disjunctive - every

finite block of integers appears in the continued fraction expansion. They pose an open question whether every computable continued fraction is random. Using technical estimates from the work of Becher and Yuhjtman [2], we show in the present work that every Martin-Löf random continued fraction is also normal, thus partially answering the open question.

We also generalize the Kolmogorov inequality for infinite-prefix free set which was proved in Lemma 4.3 in [10]. Apart from that we also prove that the gauss measure of any set  $S$  is upper bounded by  $1/k$  where there exist a supermartingale  $d$  such that  $d(w) > k$  for all  $w \in S$ .

We also define and investigate properties of a version of Kolmogorov complexity of finite continued fractions, initiating the study of individual, “random” finite continued fractions. We define the Kolmogorov complexity of the finite continued fraction as the Kolmogorov complexity of the encoded binary string representing it, which can be uniquely decoded back to the continued fraction.

## 2 Preliminaries

Let  $\mathbb{N}$  denote the set of all positive natural numbers,  $\mathbb{Q}$  denote the set of rational numbers in unit interval,  $\mathbb{N}^*$  represent the set of finite length string of natural numbers and  $N^\infty$  that of infinite sequences of natural numbers. If a string  $u \in \mathbb{N}^*$  is a prefix of some other sequence  $X \in N^\infty$  then we represent it by  $u \sqsubset X$ .  $\lambda$  represents the empty string. For any string  $w \in \mathbb{N}^*$ ,  $|w|$  represents the number of digits in the string.  $\Sigma^*$  represents the set of finite length binary strings.

Let  $a_1, a_2, \dots \in \mathbb{N}$ . We identify the sequence  $[a_1, a_2, a_3 \dots]$  with the continued fraction expansion of the real in the unit interval given by

$$\frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{a_3 + \frac{1}{\ddots}}}}. \tag{1}$$

Similarly, let  $[a_1, \dots, a_n]$  denote the finite continued fraction

$$\frac{1}{a_1 + \frac{1}{\ddots + \frac{1}{a_n}}}. \tag{2}$$

When unambiguous, we denote by the sequence  $s \in \mathbb{N}^\infty$ , the continued fraction  $[s_1, s_2, \dots]$  and by the string  $r \in \mathbb{N}^*$  the continued fraction  $[r_1, \dots, r_n]$ , where  $n$  is the number of integers in  $r$ . It is well-known that the continued fraction expansion of any real  $x$  in the unit interval is unique, and that the expansion is infinite if and only if  $x$  is irrational. (see, for example, [6]).

The cylinder set( $C_s$ ) of any string  $s \in \mathbb{N}^*$  is the set of all  $x \in (0, 1)$  whose continued fraction starts from  $s$ .

We work in the probability space  $(\mathbb{N}^\infty, \mathcal{B}(\mathbb{N}^\infty), \gamma)$  where  $\mathcal{B}(\mathbb{N}^\infty)$  is the Borel sigma algebra generated by the cylinders and  $\gamma$  is the Gauss measure defined as  $\frac{1}{\ln 2} \int_A \frac{1}{1+x} dx$  for every  $A \in \mathcal{B}(\mathbb{N}^\infty)$ . The Lebesgue measure for any  $A \in \mathcal{B}(\mathbb{N}^\infty)$  is defined as  $\int_A x dx$ . We define the left shift transformation  $T : (0, 1) \setminus \mathbb{Q} \rightarrow (0, 1) \setminus \mathbb{Q}$  by  $T(x) = (1/x) - \lfloor 1/x \rfloor$  which acts as left shift on the continued fraction expansion like  $T([a_1, a_2, a_3 \dots]) = [a_2, a_3 \dots]$ . In the same way, we define the left shift transformation  $T' : \Sigma^* \rightarrow \Sigma^*$  which acts as  $T'(b_1.b_2 \dots) = b_2.b_3, \dots$  for binary strings. For  $w \in \Sigma^*$ , we denote  $\mu(C_w)$  by  $\mu(w)$  and analogously for  $v \in \mathbb{N}^*$  and  $\gamma$ .

### 3 Definitions

**Definition 1.** A number  $X \in (0, 1) \setminus \mathbb{Q}$  is said to be continued fraction normal if for all strings  $s \in \mathbb{N}^*$

$$\lim_{n \rightarrow \infty} \frac{\#\{T^i X \in C_s : 0 \leq i < n\}}{n} = \gamma(C_s).$$

In other words, an infinite continued fraction is said to be a continued fraction normal if the asymptotic frequency of all finite length string of natural numbers is equal to the Gauss measure of its cylinder set. Note that occurrences of  $w \in \mathbb{N}^*$  within a continued fraction may partially overlap with another occurrence of  $w$ . We can also define this using the frequency of disjoint occurrences of finite strings of integers [8].

**Definition 2.** A continued fraction martingale is a function  $d : \mathbb{N}^* \rightarrow [0, \infty)$  such that  $d(\lambda) < \infty$ , and, for every  $w \in \mathbb{N}^*$ ,

$$d(w) = \sum_{i \in \mathbb{N}} d(wi)\gamma(wi|w).$$

We say that  $d : \mathbb{N}^* \rightarrow [0, \infty)$  is a continued fraction supermartingale if  $d(\lambda) < \infty$ , and the equality above is replaced with a  $\geq$ . A supermartingale or a martingale  $d$  succeeds on an infinite sequence  $X$ , if  $\limsup_{n \rightarrow \infty} d(X(1 \dots n)) = \infty$

In other words, continued fraction martingale represents the fair betting strategy which ensures that the expected value after the outcome is equal to the capital in hand before the trial. We can view  $d(w)$  as the ‘‘capital in hand’’ if  $w$  appears as an outcome. The expected value is taken with respect to the Gauss measure.

**Definition 3.** A function  $d : \mathbb{N}^* \rightarrow [0, \infty)$  is called computably enumerable (alternatively, lower semicomputable) if there exists a total computable function  $\hat{d} : \mathbb{N}^* \times \mathbb{N} \rightarrow \mathbb{Q} \cap [0, \infty)$  such that the following two conditions hold.

- **Monotonicity** : For all  $w \in \mathbb{N}^*$  and for all  $n \in \mathbb{N}$ , we have  $\hat{d}(w, n) \leq \hat{d}(w, n + 1) \leq d(w)$ .
- **Convergence** : For all  $w \in \mathbb{N}^*$ ,  $\lim_{n \rightarrow \infty} \hat{d}(w, n) = d(w)$ .

**Definition 4.** A sequence  $X \in \mathbb{N}^\infty$  is said to be Martin-Löf random if no lower semicomputable supermartingale succeeds on  $X$ .

In other words, an infinite sequence of continued fraction is said to be Martin-Löf random if no betting strategy can win infinite amount of money on the sequence with finite amount of capital in hand.

### 4 Useful Lemmas

**Lemma 1.** Let  $E \subseteq \mathbb{N}^*$  be a nonempty, computably enumerable set of finite continued fractions, with  $\gamma(E) \leq c$ , where  $0 < c < 1$ . Then there is a computably enumerable continued fraction martingale  $d : \mathbb{N}^* \rightarrow [0, \infty)$  such that for every  $w \in E$ , we have  $d(w) \geq \frac{1}{c}$ .

*Proof.* Define  $d : \mathbb{N}^* \rightarrow [0, \infty)$  by

$$d(w) = \frac{1}{c} \gamma(E|w).$$

We have  $d(\lambda) \leq 1$ . It is easy to verify that

$$\begin{aligned} \sum_{i \in \mathbb{N}} d(wi) \gamma(wi) &= \frac{1}{c} \sum_{i \in \mathbb{N}} \gamma(E|wi) \gamma(wi) \\ &= \frac{1}{c} \sum_{i \in \mathbb{N}} \gamma(E \cap wi) \\ &= \frac{1}{c} \gamma \left( E \cap \left[ \bigcup_{i \in \mathbb{N}} wi \right] \right) \\ &= \frac{1}{c} \gamma(E \cap w) \\ &= d(w) \gamma(w). \end{aligned}$$

Thus  $d$  is a martingale. Since  $E$  is computably enumerable, there is a Turing machine  $M : \mathbb{N}^* \times \mathbb{N} \rightarrow \mathbb{R}$  such that for every  $w \in \mathbb{N}^*$ ,  $M(w, n)$  monotonically converges from below to  $\gamma(E|w)$ . Hence  $d$  is lower semicomputable.

Let  $w \in E$ . Then

$$d(w) = \frac{1}{c} \gamma(E|w) = \frac{1}{c} \gamma(E \cap w) / \gamma(w) = \frac{1}{c}.$$

**Lemma 2.** (Corollary of Lemma 6 in [2]) Let  $b$  be a block of  $m$ -positive integers  $b_1, b_2 \dots b_m$ . Then for every positive real  $\delta$  and for every positive integer  $k$ ,

$$\mu \left\{ x \in C_\lambda : \left| \frac{1}{k} \sum_{i=0}^{k-1} \mathbb{I}_{[b_1, \dots, b_m]}(T^i x) - \gamma(C_b) \right| > \delta \right\} \leq 6M e^{-\frac{\delta^2 k}{2M}}$$

where  $M = M(\delta, m) = \lceil m - \log(\delta^2/2 \log 2) \rceil$ .

**Lemma 3.** (Lemma 2 in [9]) For any subinterval  $B$  of unit interval we have  $\frac{1}{2 \ln 2} \mu(B) \leq \gamma(B) \leq \frac{1}{\ln 2} \mu(B)$

**Lemma 4.** (Lemma 3.1 in [10]) For any continued fraction  $\langle a_1, a_2 \dots a_n \rangle$ ,

$$\mu(a_1, a_2 \dots a_n) \leq \frac{1}{2^n}$$

### 5 Martin-Löf Randomness and Normality

**Theorem 1.** If  $X$  is Martin-Löf random continued fraction then  $X$  is continued fraction normal.

*Proof.* We will prove the contrapositive of the above statement. Let's assume  $X$  be non-normal continued fraction. Then there are two cases:

- There exist a block  $b = b_1, b_2, \dots b_m$  such that

$$\limsup_{n \rightarrow \infty} \frac{\#\{T^i X \in C_b : 0 \leq i \leq n - 1\}}{n} > \gamma(b)$$

or

- There exists a block  $b = b_1, b_2, \dots b_m$  such that

$$\liminf_{n \rightarrow \infty} \frac{\#\{T^i X \in C_b : 0 \leq i \leq n - 1\}}{n} < \gamma(b).$$

Now, by the definition of the limit supremum and limit infimum, there exist infinitely many prefixes of  $X$  such that the frequency of the block  $b$  in  $X$  deviates from  $\gamma(b)$ . The frequency of block  $b$  in a  $k$ -length string  $K$  where  $K \geq m$  is defined as

$$\frac{\#\{T^i K \in C_b : 0 \leq i \leq (k - m)\}}{(k - m + 1)}$$

Otherwise the frequency of block  $b$  in  $k$ -length string is 0 if  $m > k$ . Now, let us consider a set of sets,  $E = \{E_1, E_2, \dots\}$  where each  $E_k$  contains all  $k$ -length continued fractions such that the frequency of  $b$  deviates from  $\gamma(b)$  in each of the continued fraction. Now from Lemma 2, it can be established that  $\mu(E_k) \leq C.e^{-k}$  and from Lemma 3,  $\gamma(E_k) \leq \frac{C.e^{-k}}{\ln 2} = C'e^{-k}$ .

Now let us consider a computably enumerable sequence of martingales  $\{d_1, d_2 \dots\}$  on  $X$  defined by,

$$\begin{aligned} d_k(\lambda) &= 1 \\ d_k(w) &= \frac{1}{C'e^{-k}} \gamma(E_k|w), & w \neq \lambda, |w| \leq k \\ d_k(wi) &= d_k(w), & |w| > k. \end{aligned}$$

From Lemma 1,  $d_k(w) \geq \frac{1}{\gamma(E_k)}$ , where  $w \sqsubset X$  and  $w \in E_k$ . Since, we are evenly betting in the martingale definition, it follows that  $\lim_{n \rightarrow \infty} d_k(X[1 \dots n]) = d_k(w)$ .

Now, consider the martingale  $d$  on  $X$ , which is a combination of  $\{d_1, d_2 \dots\}$  such that each martingale  $d_k$  bets on  $X$  as defined above with the condition that  $d_k(\lambda) = \frac{1}{2^k}$ , as the initial capital can only be finite.

Thus, for  $w \sqsubset X$  and  $w \in E_k$ ,

$$\lim_{n \rightarrow \infty} d_k(X[1 \dots n]) = \frac{d_k(w)}{2^k} = \frac{1}{C'e^{-k}2^k} > 1$$

for large enough  $k$ . As there are infinite number of prefixes of  $X$  for which the frequency of  $b$  deviates from  $\gamma(b)$ , there are infinitely many of set  $E_k$  's which contain the prefixes of  $X$ . Thus there are infinite number of martingales  $d_k$  's which make a capital greater than 1 which implies that,

$$\sum_{k \in \mathbb{N}} \lim_{n \rightarrow \infty} d_k(X[1 \dots n]) = \infty.$$

Thus, we have proved that there exist a martingale  $d$  which succeeds on  $X$  which establishes the result. □

However, the concept of a Martin-Löf random continued fraction is strictly stronger than that of a normal continued fraction. In the following result, we construct a sequence which is normal, but not Martin-Löf random.

**Theorem 2.** *There is a continued fraction  $X$  which is normal, but not Martin-Löf random.*

*Proof.* We construct an  $X$  which is continued fraction normal such that a computably enumerable martingale succeeds on it.

Consider the sequence  $X \in \mathbb{N}^\infty$  defined below.

$$X : \left\langle \frac{1}{2} \right\rangle \cdot \left\langle \frac{1}{3} \right\rangle \cdot \left\langle \frac{2}{3} \right\rangle \cdot \left\langle \frac{1}{4} \right\rangle \cdot \left\langle \frac{2}{4} \right\rangle \cdot \left\langle \frac{3}{4} \right\rangle \dots$$

where  $\left\langle \frac{p}{q} \right\rangle$  denotes the continued fraction expansion of rational number  $\frac{p}{q}$ . This is the sequence of continued fractions of the rationals enumerated in the following order: For each fixed denominator, we enumerate the numerators of proper fractions in increasing order. Then, we concatenate such rational sequences in the increasing order of denominators.

Adler et al. [1] has shown that the sequence  $X$  is continued fraction normal.

Note that the continued fraction expansion of the rational number  $(p/q)$  can be computed by running euclidean GCD algorithm on numerator and denominator which takes  $O(\log \min(p, q))$  time which is  $O(\log(p))$  in case of  $X$ . Given  $n^{th}$  rational,  $(n + 1)^{th}$  rational of the sequence  $X$  can also be computed in  $O(1)$  time.

Now, we define the function  $d$  as:

$$d(\lambda) = 1$$

$$d(wi) = \frac{d(w)}{\gamma(wi|w)},$$

$$d(wj) = 0$$

where  $i$  is the computed digit and  $j \neq i$ .

Now,

$$\sum_{i \in \mathbb{N}} d(wi) \cdot \gamma(wi) = d(wi)\gamma(wi) = d(w)\gamma(w)$$

which shows that  $d$  is a martingale.

Let us assume  $X = a_1, a_2, a_3 \dots$  where  $a_i \in \mathbb{N}$ . Now from the above martingale,

$$d(a_1, a_2 \dots a_n) = \frac{1}{\gamma(a_1, a_2 \dots a_n)}$$

From Lemma 3 and 4 we get,

$$d(a_1, a_2 \dots a_n) = \frac{1}{\gamma(a_1, a_2 \dots a_n)} \geq \frac{2 \ln 2}{\mu(a_1 \dots a_n)} \geq 2 \ln 2 \cdot 2^{-n}$$

which turns out to be  $\limsup_{n \rightarrow \infty} d(a_1, a_2 \dots a_n) = \infty$  implying computably enumerable martingale  $d$  succeeds on  $X$ .

So, we have shown that there exist a sequence which is continued fraction normal but not Martin-Löf random.

### 6 Kolmogorov Inequality

**Lemma 5.** (Lemma 4.3 in [10]) *Let  $d : \mathbb{N}^* \rightarrow [0, \infty)$  be a supermartingale and  $T \subseteq \mathbb{N}^*$  be a finite prefix-free set. Then,*

$$\sum_{w \in T} d(w)\gamma(w) \leq 1$$

**Lemma 6.** *Let  $d : \mathbb{N}^* \rightarrow [0, \infty)$  be a supermartingale and  $T \subseteq \mathbb{N}^*$  be an infinite prefix-free set. Then,*

$$\sum_{w \in T} d(w)\gamma(w) \leq 1$$

*Proof.* Let  $n \in \mathbb{N}$  be arbitrary. Let  $d_n : \mathbb{N}^* \rightarrow [0, \infty)$  be “finite-depth version” of  $d$  defined for  $w \in \mathbb{N}^*$  and  $b \in \mathbb{N}$  by :

$$d_n(wb) = \begin{cases} d(w), & \text{if } |wb| \leq n \\ 0 & \text{otherwise} \end{cases}$$



Then, it can be easily verified that  $d_n$  is a supermartingale. Now define the set  $S_n = S \cap \mathbb{N}^n$ . Then,  $S_n$  is a finite prefix-free set. We have,

$$\begin{aligned} \sum_{w \in S} d_n(w)\gamma(w) &= \sum_{w \in S_n} d_n(w)\gamma(w) \\ &= \sum_{w \in S_n} d(w)\gamma(w) \\ &\leq 1 \end{aligned}$$

where the last inequality is followed by Lemma 5 applied to  $d$  and the finite prefix-set  $S_n$ . Since,  $n$  is arbitrary and since  $d_n(w)$  is non-decreasing in  $n$  for all  $w$ , we have

$$\sum_{w \in S} d(w)\gamma(w) \leq \sup_n \sum_{w \in S_n} d_n(w)\gamma(w) \leq 1$$

as required.

**Theorem 3.** *Let  $d : \mathbb{N}^* \rightarrow [0, \infty)$  be a supermartingale and  $S \subseteq \mathbb{N}^*$  be a prefix-free set. Then for all  $k \in \mathbb{R}$ , we have*

$$\sum_{w \in S, d(w) > k} \gamma(w) \leq \frac{1}{k}$$

*Proof.* From Lemma 6 we know that,

$$\sum_{w \in S} d(w)\gamma(w) \leq 1$$

Further,

$$\begin{aligned} \sum_{w \in S} d(w)\gamma(w) &= \sum_{\substack{w \in S \\ d(w) \leq k}} d(w)\gamma(w) + \sum_{\substack{w \in S \\ d(w) > k}} d(w)\gamma(w) \\ &\geq \sum_{\substack{w \in S \\ d(w) > k}} d(w)\gamma(w) \\ &> k \sum_{\substack{w \in S \\ d(w) > k}} \gamma(w) \end{aligned}$$

Thus,

$$k \sum_{w \in S, d(w) > k} \gamma(w) \leq 1$$

as required.

## 7 A Notion of Kolmogorov Complexity for Continued Fractions

Let us assume that  $\mathcal{E}$  is some encoding which encodes the finite continued fraction into binary string such that it can be uniquely decoded back to the continued fraction. We define the Kolmogorov complexity of any finite continued fraction  $[a_1, a_2 \dots a_n]$  as  $C(\mathcal{E}[a_1, a_2 \dots a_n])$ .

Let us fix an encoding for representing the continued fraction as binary strings. Let us define a function  $bin(a)$  which returns the binary expansion of natural number  $a$ .  $X$  represents some continued fraction,  $X[i]$  denotes  $i^{th}$  digit of  $X$  and  $x$  represents the binary string.

Set  $Flag = 1$

**Encoding ( $X$ ):**

```

| if  $X = NULL$  then
|   | return 10
| end

```

```

| if  $Flag = 1$  then
|   |  $Flag = 0$ 
|   | return  $BD(bin(X[1])).Encoding(T(X))$ 
| end

```

```

| if  $Flag = 0$  then
|   | return  $01.BD(bin(X[1])).Encoding(T(X))$ 
| end

```

end

**BD ( $x$ ):**

```

| if  $x = NULL$  then
|   | return  $\lambda$ 
| end
| if  $x[1] = 1$  then
|   | return  $(11.BD(T'(x)))$ 
| end
| if  $x[1] = 0$  then
|   | return  $(00.BD(T'(x)))$ 
| end

```

end

**Algorithm 1:** Encoding of continued fraction to binary string.

The given encoding doubles the bit of binary expansion of each continued fraction digit, with 01 as separator and 10 as terminator.

**Lemma 7.** (Theorem 2(c) in [12]) For every partial computable function  $f$ , there exists a constant  $k$  such that

$$C(f(x)) \leq C(x) + k$$

**Theorem 4.** There exist a constant  $k$  such that for all binary strings  $x_1$  and  $x_2$  representing some finite continued fraction  $X$  in prefix-free encoding  $\mathcal{E}_1$  and  $\mathcal{E}_2$ ,

$$\left| C(x_1) - C(x_2) \right| \leq k$$

*Proof.* Let  $(E_1, D_1)$  and  $(E_2, D_2)$  be the Turing machines which encodes, decodes the continued fraction to binary string and vice-versa for encoding  $\mathcal{E}_1$  and  $\mathcal{E}_2$ . Let us assume  $x_1$  and  $x_2$  be binary string representing some finite continued fraction  $X$  in prefix-free encoding  $\mathcal{E}_1$  and  $\mathcal{E}_2$

Now, consider a function  $f$  as:

$$f : \Sigma^* \rightarrow \Sigma^*$$

$$f(x) = E_2(D_1(x))$$

Since, function  $f$  is defined only for those binary strings which are valid encodings in  $\mathcal{E}_1$ , it turns out that  $f$  is partial computable function. As,  $f(x_1) = x_2$ , from Lemma 7 we get there exist a constant  $k_1$ ,

$$C(x_2) \leq C(x_1) + k_1 \tag{3}$$

for all  $x_1, x_2$  representing some continued fractions in encoding  $\mathcal{E}_1$  and  $\mathcal{E}_2$ .

Now we prove the other side of the inequality. Similarly, consider a function  $g$  as :

$$g : \Sigma^* \rightarrow \Sigma^*$$

$$g(x) = E_1(D_2(x))$$

Again, function  $g$  is defined only for those binary strings which are valid encodings in  $\mathcal{E}_2$ , it turns out that  $g$  is partial computable function. As,  $f(x_2) = x_1$ , from Lemma 7 we get there exist a constant  $k_2$ ,

$$C(x_1) \leq C(x_2) + k_2 \tag{4}$$

for all  $x_1, x_2$  representing some continued fractions in encoding  $\mathcal{E}_1$  and  $\mathcal{E}_2$ . From, (3) and (4) we get,

$$\left| C(x_1) - C(x_2) \right| \leq \max(k_1, k_2) = k$$

**Acknowledgments.** The author wishes to thank Verónica Becher for the reference to their work on continued fraction normality [2] and to Satyadev Nandakumar for helpful discussions.

## References

1. Adler, R., Keane, M., Smorodinsky, M.: A construction of a normal number for the continued fraction transformation. *J. Number Theory* **13**(1), 95–105 (1981)
2. Becher, V., Yuhjtman, S.A.: On absolutely normal and continued fraction normal numbers. *Int. Math. Res. Not.* **2019**(19), 6136–6161 (2018)
3. Borel, É.: Sur les probabilités dénombrables et leurs applications arithmétiques. *Rend. Circ. Mat. Palermo* **27**, 247–271 (1909)
4. Dajani, K., Kraaikamp, C.: *Ergodic Theory of Numbers*. The Mathematical Association of America (2002)
5. Einsiedler, M., Ward, T.: *Ergodic Theory: With a View Towards Number Theory*. Graduate Texts in Mathematics, Springer, London (2010). <https://doi.org/10.1007/978-0-85729-021-2>
6. Khinchin, A.Y.: *Continued Fractions*. Dover Publications, New York (1997)
7. Li, M., Vitanyi, P.: *An Introduction to Kolmogorov Complexity and Its Applications*, 4th edn. Springer, New York (2019). <https://doi.org/10.1007/978-0-387-49820-1>
8. Nandakumar, S., Pulari, S., Vishnoi, P., Viswanathan, G.: An analogue of Pillai’s theorem for continued fraction normality and an application to subsequences. *Bull. Lond. Math. Soc.* **53**(5), 1414–1428 (2021)
9. Nandakumar, S., Vishnoi, P.: Randomness and effective dimension of continued fractions. In: Esparza, J., Kráľ, D. (eds.) *45th International Symposium on Mathematical Foundations of Computer Science (MFCS 2020)*, Leibniz International Proceedings in Informatics (LIPIcs), Dagstuhl, Germany, vol. 170, pp. 73:1–73:13. Schloss Dagstuhl-Leibniz-Zentrum für Informatik (2020)
10. Nandakumar, S., Vishnoi, P.: On continued fraction randomness and normality. *Inf. Comput.* 104876 (2022)
11. Scheerer, A.-M.: On the continued fraction expansion of absolutely normal numbers. *J. Th. Nombres Bordeaux* (2017, to appear)
12. Shen, A.: Around Kolmogorov complexity: basic notions and results. CoRR, abs/1504.04955 (2015)
13. Vandehey, J.: Absolutely abnormal and continued fraction normal numbers. *Bull. Aust. Math. Soc.* **94**(2) (2016)



# Weakly $k$ -submodular Maximization Under Matroid Constraint

Yijing Wang<sup>1</sup>, Dongmei Zhang<sup>2</sup>, Yapu Zhang<sup>3(✉)</sup>, and Zhenning Zhang<sup>3</sup>

<sup>1</sup> Academy of Mathematics and Systems Science, Chinese Academy of Sciences,  
Beijing 100190, People's Republic of China

<sup>2</sup> School of Computer Science and Technology, Shandong Jianzhu University,  
Jinan 250101, People's Republic of China

<sup>3</sup> Beijing Institute for Scientific and Engineering Computing,  
Beijing University of Technology, Beijing 100124, People's Republic of China  
zhangyapu@bjut.edu.cn

**Abstract.** In this work, we investigate the problem that maximizes a weakly  $k$ -submodular function under the matroid constraint. Different from traditional submodular function maximization, there are  $k$  disjoint subsets in  $k$ -submodular function optimization, instead of a single set in the submodular maximization. For the weakly  $k$ -submodular maximization problem, we provide a greedy algorithm whose approximation ratio is  $\alpha/(1 + \alpha)$ , where parameter  $0 < \alpha \leq 1$  is the orthant submodularity ratio. Then we extend to cardinality constraint which maintains the same performance ratio.

**Keywords:** Weak function · Matroid constraint · Cardinality constraint ·  $k$ -submodular · Approximation algorithms

## 1 Introduction

Submodular optimization occupies great importance in combinatorial optimization, as well as in many applications and real life. The  $k$ -submodular optimization is a natural extension of submodular optimization, which integrates multiple research directions such as machine learning, graph theory, combinatorial optimization and game theory, etc. It is an important embodiment of promoting the development for cross-disciplines. The problem  $k$ -submodular maximization makes great contribution in multi-sensor placement, multi-topic influence maximization propagation, multi-type subset selection, and other forms of web page recommendation [5, 7, 8, 10].

Instead of a single subset,  $k$ -submodular function aims to find  $k$  disjoint subsets to express submodularity or orthant submodularity. For  $k$ -submodular maximization, function  $f$  is traditional submodular when  $k$  takes 1, and bisubmodular when  $k$  takes 2. For bisubmodular function, Singh et al. [9] proposed an approximation algorithm for maximization model. For the problem maximizing a  $k$ -submodular function without any constraints, Ward and Živný [12] firstly

designed an algorithm which gets a  $1/(1 + \sqrt{k/2})$  ratio. When function  $f$  is monotone, Iwata et al. [2] proposed a  $k/(2k-1)$ -approximation algorithm; when function  $f$  is nonmonotone, they provided a random approximation algorithm, whose ratio is  $1/2$ . Oshima [6] improved the ratio to  $(k^2+1)/(2k^2+1)$  by utilizing randomized technique for the same case. Iwata et al. [2] gave an algorithm whose ratio is better than  $(k+1)/2k$  under exponential queries. Soma [10] showed there does not exist  $1/2$ -regret and  $k/(2k-1)$ -regret algorithms for nonmonotone case and monotone case respectively under online model.

Depending on different application scenarios, there are a variety of constraints for  $k$ -submodular maximization, including matroid, cardinality, and knapsack. For cardinality constraints, there are total cardinality constraint and individual cardinality constraint respectively. For total size case, Yoshida [5] showed an approximation algorithm by greedy technique whose performance guarantee is  $1/2$ ; and for individual size case, the approximation ratio is  $1/3$ . For matroid constraint, Sakaue [8] showed an approximation algorithm which obtains the same performance as Yoshida [5]. For knapsack constraint, Tang et al. [11] got an algorithm taking an approximation ratio of  $(1/2 - 1/2e)$ . Qian et al. [7] also showed  $1/2$  performance guarantee for the total size case in polynomial time by means of the multi-objective evolutionary technique.

There are still many objective functions that are not strictly  $k$ -submodular [3, 4, 14]. Zheng et al. [14] designed an approximation algorithm for  $\epsilon$ -approximately  $k$ -submodular maximization under individual size case, and the performance guarantee is  $(1 - \epsilon)/(3 + \epsilon)$ . Nguyen and Thai [4] showed an approximation algorithm with respect to the function structure for  $\epsilon$ -approximately  $k$ -submodular maximization. Matsuoka and Ohsaka [3] provided two approximation algorithms for maximizing a  $\gamma$ -weakly  $k$ -submodular function under the constraints of matroid and individual cardinality respectively, and the performance ratios are both  $(1 + 1/\gamma)^{-2}$ .

In this paper, we research the problem that maximizes a weakly  $k$ -submodular function with matroid constraint. Utilizing greedy technique, we design an approximation algorithm for the optimization model, whose approximation ratio is  $\alpha/(1 + \alpha)$ , where parameter  $0 < \alpha \leq 1$ . Next we extend to a special case, i.e., cardinality constraint, which maintains the same approximation ratio.

The rest of this work consists of four parts. In Sect. 2, we introduce some definitions and notations including weakly  $k$ -submodular function. In Sect. 3, we provide an algorithm for weakly  $k$ -submodular maximization problem under matroid constraint. In Sect. 4, we show a special algorithm for maximizing a weakly  $k$ -submodular function subject to cardinality constraint. We remark a conclusion in Sect. 5.

## 2 Preliminaries

In this work, we investigate the problem that maximizes a set function  $f$  satisfying weakly  $k$ -submodularity with a matroid constraint, which can be formulated as:

$$\begin{aligned} \max \quad & f(P) \\ \text{s.t.} \quad & P = \{P_1 \cup P_2 \cup \dots \cup P_k\} \in \mathcal{I}, \end{aligned}$$

where  $\mathcal{I}$  contains all independent sets of matroid  $\mathcal{M} = (V, \mathcal{I})$ , and function  $f : (k + 1)^V \rightarrow \mathbb{R}_+$  maintains weakly  $k$ -submodularity over  $k$  disjoint subsets of the ground set  $V = \{v_1, v_2, \dots, v_n\}$ . Next we will introduce some preliminaries about the model in detail.

Firstly, we give the definition of submodular function. For the ground set  $V = \{v_1, v_2, \dots, v_n\}$  and its traditional subset family  $2^V$ , set function  $g : 2^V \rightarrow \mathbb{R}$  is submodular if  $g$  satisfies that  $g(W_1) + g(W_2) \geq g(W_1 \cup W_2) + g(W_1 \cap W_2)$ , for any subsets  $W_1, W_2 \subseteq V$ . In submodular maximization problem, we usually require the set function  $g : 2^V \rightarrow \mathbb{R}$  satisfies monotonicity and normalization, as well as non-negativity. Here we propose some relevant introductions about monotonicity, normalization and non-negativity for function  $g$ .

- monotonicity: for subsets  $P_1, P_2 \subseteq V$ , if  $P_1 \subseteq P_2$ , then  $g(P_1) \leq g(P_2)$ .
- normalization: for the empty set  $\emptyset$ ,  $g(\emptyset) = 0$ .
- non-negativity: for any subset  $P \subseteq V$ ,  $g(P) \geq 0$ .

Different from the traditional submodular function, Huber and Kolmogorov [1] provided a definition of  $k$ -submodular function. For  $k$ -submodular functions, they maintain submodularity over  $k$  disjoint sets rather than a single set. In this case, denote  $(k + 1)^V := \{(P_1, P_2, \dots, P_k) | P_i \subseteq V (i = 1, 2, \dots, k), P_i \cap P_j = \emptyset (i \neq j)\}$ . For the sake of expression convenience, we denote vector  $\mathbf{p} = \{(P_1, P_2, \dots, P_k)\}$ . The detailed description for  $k$ -submodular function can be found as follows.

**Definition 1 ( $k$ -submodular function).** Set function  $f : (k + 1)^V \rightarrow \mathbb{R}$  is  $k$ -submodular if it satisfies  $f(\mathbf{p}) + f(\mathbf{q}) \geq f(\mathbf{p} \sqcap \mathbf{q}) + f(\mathbf{p} \sqcup \mathbf{q})$ , where  $\mathbf{p} = (P_1, P_2, \dots, P_k)$ ,  $\mathbf{q} = (Q_1, Q_2, \dots, Q_k)$ ,

$$\mathbf{p} \sqcap \mathbf{q} := (P_1 \cap Q_1, P_2 \cap Q_2, \dots, P_i \cap Q_i, \dots, P_k \cap Q_k),$$

$$\mathbf{p} \sqcup \mathbf{q} := \left( P_1 \cup Q_1 \setminus \left( \bigcup_{i \neq 1} P_i \cup Q_i \right), \dots, P_k \cup Q_k \setminus \left( \bigcup_{i \neq k} P_i \cup Q_i \right) \right),$$

for the sets in feasible region

$$(k + 1)^V := \{(Q_1, Q_2, \dots, Q_k) : Q_i \subseteq V (i = 1, 2, \dots, k), Q_i \cap Q_j = \emptyset (i \neq j)\}.$$

For the  $k$ -submodular maximization problem, let  $[k] := \{1, 2, \dots, k\}$ . For two vectors  $\mathbf{p} = (P_1, P_2, \dots, P_k)$ ,  $\mathbf{q} = (Q_1, Q_2, \dots, Q_k)$  in  $(k + 1)^V$ , let “ $\preceq$ ” be a partial order such that  $\mathbf{p} \preceq \mathbf{q}$  if for any  $i \in [k]$ , there is  $P_i \subseteq Q_i$ . For partial order vectors  $\mathbf{p} = (P_1, P_2, \dots, P_k)$ ,  $\mathbf{q} = (Q_1, Q_2, \dots, Q_k)$ ,  $\mathbf{p} \prec \mathbf{q}$ , if there is some  $i \in [k]$ , such that  $P_i \subset Q_i$ , we call it a strongly partial order, i.e.,  $\mathbf{p} \prec \mathbf{q}$ .

As for  $k$ -submodular function, we denote

$$f(v, i|\mathbf{p}) := f(P_1, P_2, \dots, P_i \cup \{v\}, \dots, P_k) - f(P_1, P_2, \dots, P_i, \dots, P_k)$$

be the marginal gain while we add element  $v$  to the  $i$ -th subset of vector  $\mathbf{p}$  for  $\mathbf{p} \in (k + 1)^V$ ,  $v \notin \cup_{l \in [k]} P_l$ ,  $i = 1, 2, \dots, k$ . We call  $k$ -submodular function  $f$  is monotone if  $f(v, i|\mathbf{p}) \geq 0$  for all vectors  $\mathbf{p} \in (k + 1)^V$ ,  $v \notin \cup_{l \in [k]} P_l$  and  $i = 1, 2, \dots, k$ . Motivated by the work of Ward and Živný [13], we propose an introduction for weakly  $k$ -submodular function, which can be found in Definition 2.

**Definition 2 (Weakly  $k$ -submodular function).** *Function  $f : (k + 1)^V \rightarrow \mathbb{R}$  is weakly  $k$ -submodular if  $f(v, i|\mathbf{p}) \geq \alpha \cdot f(v, i|\mathbf{q})$ , for any vectors  $\mathbf{p}, \mathbf{q}$  in  $(k + 1)^V$  with  $\mathbf{p} \preceq \mathbf{q}$ ,  $v \notin \cup_{l \in [k]} Q_l$  and  $i = 1, 2, \dots, k$ ,  $0 < \alpha \leq 1$ .*

When  $\alpha = 1$ , function  $f$  satisfies orthant submodularity proposed by Ward and Živný [13]. In this work, we call function  $f$  weakly  $k$ -orthant submodular as weakly  $k$ -submodular function for short. For vector  $\mathbf{p}$  in  $(k + 1)^V$ , we define  $P_i = \{v \in V | \mathbf{p}(v) = i\}$  for  $i = 1, 2, \dots, k$ , and  $\text{supp}(\mathbf{p}) := \{v \in V | \mathbf{p}(v) \neq 0\}$ . The cardinality of vector  $\mathbf{p}$  can be regarded as  $|\text{supp}(\mathbf{p})|$ . Denote  $\mathbf{0}$  be the zero vector in  $(k + 1)^V$ , and assume function  $f$  is normalized, i.e.,  $f(\mathbf{0})$  equals to 0. If  $f(\mathbf{0})$  is not 0, we reset  $f(\mathbf{p}) := f(\mathbf{p}) - f(\mathbf{0})$  for any vector  $\mathbf{p}$  in  $(k + 1)^V$ .

In real life applications, there are a variety of constraint scenarios, such as cardinality constraints, knapsack constraints, matroid constraints, box constraints, and so on. A cardinality constraint requires that the cardinality of a selected subset is limited to a certain number. In knapsack constraints, every element has a weight and the total weight of the chosen subset is limited to a certain condition. Matroid is defined based on independent systems. The specific definition is as follows.

**Definition 3 (Independent system).** *For the ground set  $V = \{v_1, v_2, \dots, v_n\}$  and a subset family  $\mathcal{I}$  in  $V$ ,  $\mathcal{M} = (V, \mathcal{I})$  is an independent system such that*

- $\emptyset \in \mathcal{I}$ .
- if  $Q_1 \subseteq Q_2 \in \mathcal{I}$ , then  $Q_1 \in \mathcal{I}$ .

If a subset  $P \in \mathcal{I}$ ,  $P$  is independent. Based on the definition of independent system, independent system  $\mathcal{M} = (V, \mathcal{I})$  is a matroid if for any subsets  $P_1, P_2 \in \mathcal{I}$  and the cardinality of  $P_1$  is less than that in  $P_2$ , then there exists element  $v \in P_2 \setminus P_1$  such that  $P_1 \cup \{v\}$  is independent. For the matroid  $\mathcal{M} = (V, \mathcal{I})$ , if independent set  $P$  has the maximal size, i.e.,  $P$  is a maximal independent set, we call  $P$  is a base of the matroid  $\mathcal{M}$ . Let  $\mathcal{B}$  be the subset family containing all bases and the cardinality of a base is  $B$ . All the bases have the same cardinality observably.

We assume there exist two oracles  $\mathcal{O}_E, \mathcal{O}_I$  for function evaluation and independent sets examination respectively. The number of oracle queries express the time complexity.



### 3 Maximizing a Weakly $k$ -submodular Function with Matroid Constraint

In this section, we design a greedy algorithm for maximizing a weakly  $k$ -submodular function ( $k$ -SM for short) with matroid constraint. We choose an appropriate pair  $(v, i)$  which takes the maximal marginal gain to the current vector iteratively if it satisfies the matroid constraint. The detailed description is as follows.

---

**Algorithm 1.** A greedy algorithm for weakly  $k$ -SM under matroid

---

**Input:** a weakly  $k$ -submodular function  $f : (k + 1)^V \rightarrow \mathbb{R}_+$ , a matroid  $\mathcal{M} = (V, \mathcal{I})$ , and two oracles  $\mathcal{O}_E, \mathcal{O}_I$  for function evaluation and independence identify respectively.

**Output:** a vector  $\mathbf{v}$  satisfying  $\text{supp}(\mathbf{v})$  is a base.

**Step 0.** Initially set  $\mathbf{v} = \mathbf{0}$ ,  $R := V$ .

**Step 1.** Select element  $v \in R$  such that  $\text{supp}(\mathbf{v}) \cup \{v\} \in \mathcal{I}$  and

$$(v, i) := \arg \max_{v \in R, i \in [k]} f(v, v(i) | \mathbf{v}),$$

update  $\mathbf{v} := \mathbf{v} \cup (v, i)$ ,

remove elements  $v'$  from  $R$  such that  $\text{supp}(\mathbf{v}) \cup \{v'\} \notin \mathcal{I}$ .

**Step 2.** Repeat Step 1 until there is no elements  $v$  in  $R$  satisfying the conditions of Step 1, output  $\mathbf{v}$ .

---

By the monotonicity of function  $f$ , we know the support set of output vector  $\mathbf{v}$  is a base, i.e.,  $\text{supp}(\mathbf{v}) \in \mathcal{B}$ . The detailed proof can be found in [8].

**Lemma 4** (Sakaue [8]). *The support set of any maximal optimal solution is a base for the optimization model.*

According to the properties of bases and independent sets in matroid  $\mathcal{M} = (V, \mathcal{I})$ , the following lemma holds. Sakaue [8] provided a detailed proof for the lemma. It is recommended to the interested readers.

**Lemma 5** (Sakaue [8]). *Assume  $P$  is an independent set and  $Q$  is a base in matroid  $\mathcal{M} = (V, \mathcal{I})$ ,  $P \subset Q$ . We get that, for any element  $p \in V \setminus P$  such that  $P \cup \{p\} \in \mathcal{I}$ , there is element  $q \in Q \setminus P$  satisfying  $Q \setminus \{q\} \cup \{p\}$  is a base.*

**Theorem 6.** *For the optimization model under matroid constraint, Algorithm 1 outputs a vector  $\mathbf{v}$  such that  $f(\mathbf{v}) \geq \frac{\alpha}{1+\alpha} f(\mathbf{v}^*)$ , where  $\mathbf{v}^*$  is a maximal optimal solution and parameter  $\alpha \in (0, 1]$ . The time complexity is  $O(Bn(\mathcal{O}_I + k\mathcal{O}_E))$ .*

From Algorithm 1, we know there are totally  $B$  iterations. Among the iterations, we assume  $\mathbf{v}_j$  is the output vector after  $j$ -th iteration,  $j = 1, 2, \dots, B$

and  $\mathbf{v}^*$  is a maximal optimal solution for the optimization model. For the purpose of completing the proof of Theorem 6, we define a vector sequence  $\mathbf{v}_0^*, \mathbf{v}_1^*, \mathbf{v}_2^*, \dots, \mathbf{v}_B^*$  for the optimal solution  $\mathbf{v}^*$  as those in the works of [2, 5, 13].

Denote  $(v_j, i(j))$  be the pair selected by Algorithm 1 at the  $j$ -th iteration,  $V_j := \text{supp}(\mathbf{v}_j)$ ,  $j = 1, 2, \dots, B$ . For a maximal optimal solution  $\mathbf{v}^*$ , we construct a vector sequence  $\mathbf{v}_0^* = \mathbf{v}^*, \mathbf{v}_1^*, \mathbf{v}_2^*, \dots, \mathbf{v}_B^* = \mathbf{v}$  satisfying the following properties, where  $V_j^* := \text{supp}(\mathbf{v}_j^*)$  for  $j = 1, 2, \dots, B$ .

$$(i) \quad \mathbf{v}_j \prec \mathbf{v}_j^* \quad \text{if } j = 0, 1, 2, \dots, B - 1, \quad \text{and } \mathbf{v}_j \prec \mathbf{v}_j^* = \mathbf{v}, \quad \text{if } j = B;$$

$$(ii) \quad V_j^* \in \mathcal{B} \quad \text{for } j = 0, 1, 2, \dots, B.$$

We construct the vector sequence inductively. Firstly, we note that  $\mathbf{v}_0 = \mathbf{0}$ , and  $\mathbf{v}_0^* = \mathbf{v}^*$  satisfy (i) and (ii). Next we show how to get  $\mathbf{v}_j^*$  from  $\mathbf{v}_{j-1}^*$ , given the assumption  $\mathbf{v}_{j-1} \prec \mathbf{v}_{j-1}^*$  and  $V_{j-1}^* \in \mathcal{B}$ . As  $\mathbf{v}_{j-1} \prec \mathbf{v}_{j-1}^*$ , we know  $V_{j-1} \subset V_{j-1}^*$  and element  $v_j$  is selected to satisfy  $V_{j-1} \cup \{v_j\} \in \mathcal{I}$ . According to Lemma 5, we have there is element  $v' \in V_{j-1}^* \setminus V_{j-1}$  such that  $V_{j-1}^* \setminus \{v'\} \cup \{v_j\}$  is a base. We let  $v_j^* = v'$  and denote  $\mathbf{v}_{j-1/2}^*$  be an immediate vector obtained by setting 0 to the  $v_j^*$ -th element in subset  $V_{j-1}^*$ . Denote  $\mathbf{v}_j^*$  be the vector returned by setting  $i(j)$  to the  $v_j$ -th element from  $\mathbf{v}_{j-1/2}^*$ . Then the vector  $\mathbf{v}_j^*$  is constructed such that  $V_j^* = \{V_{j-1}^* \setminus \{v_j^*\}\} \cup \{v_j\} \in \mathcal{B}$ .

Moreover, since the immediate vector  $\mathbf{v}_{j-1/2}^*$  satisfies  $\mathbf{v}_{j-1} \preceq \mathbf{v}_{j-1/2}^*$ , we obtain the properties for  $\mathbf{v}_j^*$  as follows:

$$\mathbf{v}_j \prec \mathbf{v}_j^* \quad \text{if } j = 1, 2, \dots, B - 1, \quad \text{and } \mathbf{v}_j \prec \mathbf{v}_j^* = \mathbf{v}, \quad \text{if } j = B,$$

where the strict partial order of the inclusion can be verified by the relation  $|V_j| = j < B = |V_j^*|$  for  $j = 1, 2, \dots, B - 1$ . Therefore, utilizing the above discussion process iteratively for  $j = 1, 2, \dots, B$ , we acquire the vector sequence  $\mathbf{v}_0^*, \mathbf{v}_1^*, \mathbf{v}_2^*, \dots, \mathbf{v}_B^*$  meet the conditions (i) and (ii). In the following work, we turn to the analysis of Algorithm 1.

*Proof.* Based on the properties of vector sequence  $\mathbf{v}_0^*, \mathbf{v}_1^*, \mathbf{v}_2^*, \dots, \mathbf{v}_B^*$  constructed above, we know  $V_{j-1} \subset V_{j-1}^*$  and  $v_j^* \in V_{j-1}^* \setminus V_{j-1}$ , as well as  $V_{j-1} \cup \{v_j^*\} \subseteq V_{j-1}^* \in \mathcal{B}$ , for  $j = 1, 2, \dots, B$ . According to the down-closed property of matroid  $\mathcal{M} = (V, \mathcal{I})$ , we acquire  $V_{j-1} \cup \{v_j^*\} \in \mathcal{I}$  for  $j = 1, 2, \dots, B$ .

Given the element selection rule in Algorithm 1, for the chosen pair  $(v_j, i(j))$ , we obtain

$$f(v_j, i(j)|\mathbf{v}_{j-1}) \geq f(v_j^*, \mathbf{v}_{j-1}^*(v_j^*)|\mathbf{v}_{j-1}). \tag{1}$$

Furthermore, based on the definitions of weakly orthant submodularity and the inequality  $\mathbf{v}_{j-1} \prec \mathbf{v}_{j-1/2}^*$ , there is

$$f(v_j^*, \mathbf{v}_{j-1}^*(v_j^*)|\mathbf{v}_{j-1}) \geq \alpha \cdot f(v_j^*, \mathbf{v}_{j-1}^*(v_j^*)|\mathbf{v}_{j-1/2}^*), \quad \alpha \in (0, 1]. \tag{2}$$

Combining Inequality (1) and (2) with monotonicity of function  $f$ , we obtain

$$\begin{aligned}
 f(\mathbf{v}_j) - f(\mathbf{v}_{j-1}) &= f(v_j, i(j)|\mathbf{v}_{j-1}) \\
 &\geq f(v_j^*, \mathbf{v}_{j-1}^*(v_j^*)|\mathbf{v}_{j-1}) \\
 &\geq \alpha \cdot f(v_j^*, \mathbf{v}_{j-1}^*(v_j^*)|\mathbf{v}_{j-1/2}^*) \\
 &\geq \alpha \cdot f(v_j^*, \mathbf{v}_{j-1}^*(v_j^*)|\mathbf{v}_{j-1/2}^*) - \alpha \cdot f(v_j, i(j)|\mathbf{v}_{j-1/2}^*) \\
 &= \alpha \cdot (f(v_j^*, \mathbf{v}_{j-1}^*(v_j^*)|\mathbf{v}_{j-1/2}^*) - f(v_j, i(j)|\mathbf{v}_{j-1/2}^*)) \\
 &= \alpha \cdot (f(\mathbf{v}_{j-1}^*) - f(\mathbf{v}_j^*)).
 \end{aligned}$$

Rearranging above inequalities, we have

$$f(\mathbf{v}_{j-1}^*) - f(\mathbf{v}_j^*) \leq \frac{1}{\alpha} \cdot (f(\mathbf{v}_j) - f(\mathbf{v}_{j-1})). \quad (3)$$

Based on Inequality (3), we get

$$\begin{aligned}
 f(\mathbf{v}^*) - f(\mathbf{v}) &= \sum_{j=1}^B (f(\mathbf{v}_{j-1}^*) - f(\mathbf{v}_j^*)) \\
 &\leq \frac{1}{\alpha} \cdot \sum_{j=1}^B (f(\mathbf{v}_j) - f(\mathbf{v}_{j-1})) \\
 &= \frac{1}{\alpha} \cdot (f(\mathbf{v}) - f(\mathbf{0})) \\
 &= \frac{1}{\alpha} \cdot f(\mathbf{v}).
 \end{aligned}$$

Therefore, there is  $f(\mathbf{v}) \geq \frac{\alpha}{1+\alpha} f(\mathbf{v}^*)$  for the output vector  $\mathbf{v}$  in Algorithm 1.

Next, we express the analysis for the time complexity in Algorithm 1. In the  $j$ -th iteration, we need to scan  $R$  elements to examine the independence and call for  $k \cdot R$  evaluation oracle  $\mathcal{O}_E$  to select the maximal pair  $(v_j, i(j))$ . There are totally  $B$  iterations. Thus the complexity in Algorithm 1 is  $O(B \cdot n \cdot (\mathcal{O}_I + k\mathcal{O}_E))$ .  $\square$

## 4 Maximizing a Weakly $k$ -submodular Function with Cardinality Constraint

In this part, we study a special case of matroid constraint, i.e., cardinality constraint. More specifically, we investigate the problem that maximizes the weakly  $k$ -submodular function  $f : (k+1)^V \rightarrow \mathbb{R}_+$  such that the cardinality of  $\text{supp}(\mathbf{v})$  is no more than a positive integer  $C$ . For this problem, we provide Algorithm 2 to find a vector  $\mathbf{v}$ .

By analyzing Algorithm 2, we acquire Theorem 7, whose main proof idea is similar to that in Theorem 6 based on the fact size constraint is a uniform matroid constraint.

---

**Algorithm 2.** A greedy algorithm for weakly  $k$ -SM under cardinality

---

**Input:** a weakly  $k$ -submodular function  $f : (k + 1)^V \rightarrow \mathbb{R}_+$ , a cardinality constraint  $C$ , and an evaluation oracle  $\mathcal{O}_E$ .

**Output:** a vector  $\mathbf{v}$  satisfying  $|\text{supp}(\mathbf{v})| \leq C$ .

**Step 0.** Initially set  $\mathbf{v} = \mathbf{0}$ ,  $R := V$ .

**Step 1.** Select element  $v \in R$  such that  $|\text{supp}(\mathbf{v})| < C$  and

$$(v, i) := \arg \max_{v \in R, i \in [k]} f(v, v(i)|\mathbf{v}),$$

update  $\mathbf{v} := \mathbf{v} \cup (v, i)$ ,  $R := R \setminus \{v\}$ .

**Step 2.** Repeat Step 1 until there is no elements  $v$  in  $R$  satisfying the conditions of Step 1, output  $\mathbf{v}$ .

---

**Theorem 7.** For the optimization model with cardinality constraint, Algorithm 2 outputs a vector  $\mathbf{v}$  satisfying  $f(\mathbf{v}) \geq \frac{\alpha}{1+\alpha} f(\mathbf{v}^*)$ , where  $\mathbf{v}^*$  is a maximal optimal solution and parameter  $\alpha \in (0, 1]$ . The complexity is  $O(C \cdot n \cdot k\mathcal{O}_E)$ .

## 5 Conclusions

In this work, we investigate the problem that maximizes a function meeting weakly  $k$ -submodularity subject to matroid constraint. By means of greedy technique, we provide an approximation algorithm, whose performance guarantee is  $\alpha/(1 + \alpha)$ , where  $\alpha \in (0, 1]$  is the ratio to characterize orthant submodularity. Next we extend to a special case, i.e., cardinality constraint, which maintains the same approximation ratio. In future work, we will further study weakly  $k$ -submodular maximization under streaming and online model.

**Acknowledgements.** The first author is supported by National Natural Science Foundation of China (Nos. 72192804, 72192800) and the Guozhi Xu Postdoctoral Research Foundation. The second author is supported by National Natural Science Foundation of China (No. 11871081). The fourth author is supported by National Natural Science Foundation of China (Nos. 12131003, 12001025).

## References

1. Huber, A., Kolmogorov, V.: Towards minimizing  $k$ -submodular functions. In: Proceedings of International Conference on Intelligent Systems and Control (ISCO), pp. 451–462 (2012)
2. Iwata, S., Tanigawa, S., Yoshida, Y.: Improved approximation algorithms for  $k$ -submodular function maximization. In: Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 404–413 (2016)
3. Matsuoka, T., Ohsaka, N.: Maximization of monotone  $k$ -submodular functions with bounded curvature and non- $k$ -submodular functions. In: Proceedings of Asian Conference on Machine Learning (ACML). vol. 157 (2021)

4. Nguyen, L., Thai, M.T.: Streaming  $k$ -submodular maximization under noise subject to size constraint. In: Proceedings of the Thirty-Seventh International Conference on Machine Learning (ICML), pp. 7338–7347 (2020)
5. Ohsaka, N., Yoshida, Y.: Monotone  $k$ -submodular function maximization with size constraints. In: Proceedings of Advances in Neural Information Processing Systems (NIPS), pp. 694–702 (2015)
6. Oshima, H.: Improved randomized algorithm for  $k$ -submodular function maximization. *SIAM J. Discrete Math.* **35**(1), 1–22 (2021)
7. Qian, C., Shi, J.C., Tang, K., Zhou, Z.H.: Constrained monotone  $k$ -submodular function maximization using multiobjective evolutionary algorithms with theoretical guarantee. *IEEE Trans. Evol. Comput.* **22**, 595–608 (2017)
8. Sakaue, S.: On maximizing a monotone  $k$ -submodular function subject to a matroid constraint. *Discrete Optim.* **23**, 105–113 (2017)
9. Singh, A., Guillory, A., Bilmes, J.: On bisubmodular maximization. In: Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics (AISTATS), pp. 1055–1063 (2012)
10. Soma, T.: No-regret algorithms for online  $k$ -submodular maximization. In: Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics (AISTATS), pp. 1205–1214 (2019)
11. Tang, Z., Wang, C., Chan, H.: On maximizing a monotone  $k$ -submodular function under a knapsack constraint. *Oper. Res. Lett.* **50**, 28–31 (2022)
12. Ward, J., Živný, S.: Maximizing bisubmodular and  $k$ -submodular functions. In: Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 1468–1481 (2014)
13. Ward, J., Živný, S.: Maximizing  $k$ -submodular functions and beyond. *ACM Trans. Algorithms* **12**(4), 1–26 (2016)
14. Zheng, L., Chan, H., Loukides, G., Li, M.: Maximizing approximately  $k$ -submodular functions. In: Proceedings of the 2021 SIAM International Conference on Data Mining (SDM), pp. 414–422 (2021)



# Approximation Algorithms for Diversity-Bounded Center Problems

Lu Han<sup>1</sup>, Shuilian Liu<sup>2</sup>, Yicheng Xu<sup>2(✉)</sup>, and Yong Zhang<sup>2</sup>

<sup>1</sup> School of Science, Beijing University of Posts and Telecommunications,  
Beijing 100876, People's Republic of China

hl@bupt.edu.cn

<sup>2</sup> Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences,  
Shenzhen 518055, People's Republic of China

{sl.liu1, yc.xu, zhangyong}@siat.ac.cn

**Abstract.** This paper considers the diversity-bounded center problems, where we are given a set of points, each of which was colored in one of the  $\omega$  colors, along with integers  $k$  and  $l_i, u_i$  for color  $i$ , the goal is to select a  $k$ -sized center set so as to minimize the maximum distance of a point to its nearest center, and at the same time, meet the requirements that the amount of selected centers with color  $i$  must be within  $[l_i, u_i]$  for each  $i$ . The diversity-bounded clustering with one-side upper bound and lower bound requirement was considered in (Jones et al., 2020) and (Thejaswi et al., 2021), respectively. We combine the difficulties of them and propose the diversity-bounded center problems from both sides, and as the main contribution, we present 3-approximation algorithms for the red-blue as well as the multi-colored version, the complexity of which for the latter problem is parameterized by  $\omega$ .

**Keywords:** Fair clustering · Diversity-bounded center ·  
Approximation algorithm · Parameterized approximation

## 1 Introduction

In the classical  $k$ -center problem, we are given a set of points  $V$  in a metric space, along with an integer  $k$ . The goal is to find a  $k$ -sized subset  $O \subseteq V$  so as to minimize the maximum distance from a point in  $V$  to its nearest center in  $O$ . The  $k$ -center problem is no doubt one of the most well-studied NP-hard problems in the combinatorial optimization literature. It is known that the classical  $k$ -center problem is even NP-hard to approximate the optimal value within a factor of  $2 - \varepsilon$ , but however, a 2-approximation algorithm can be easily obtained via a simple greedy algorithm [7]. The greedy scheme starts with an arbitrary point

---

L. Han—Supported by the National Natural Science Foundation of China (No. 12001523). Shuilian Liu and Yicheng Xu are supported by the Fundamental Research Project of Shenzhen City (No. JCYJ20210324102012033). Yong Zhang is supported by the National Natural Science Foundation of China (No. 12071460).

and iteratively select the farthest point from the current selected center set as the next center until it reaches the cardinality factor  $k$ .

In the *diversity-bounded multi-colored center* (DB-MC-Center) problem, given are a set of points  $V$  in a metric space along with an integer  $k$ , where each point is labeled with one of the  $\omega$  colors. In order to guarantee the fairness of a clustering scheme, we request that the number of selected centers should be no more than  $k$ , and satisfy given upper and lower bound constraints for each color. The DB-MC-Center problem we consider is inspired by the recent trend of fairness investigation in clustering, which was first introduced by Chierichetti et al. [5]. A fair clustering, according to their definition, is a clustering where every demographic group is approximately proportionally represented within each cluster. Without confusion, we denote the *diversity-bounded red-blue center* (DB-RB-Center) problem as the two-color case (*w.l.o.g.* we assume red and blue) of the DB-MC-Center problem.

To the best of our knowledge, the DB-MC-Center problem described above is new in the fair clustering literature, but several related models or special cases were considered in previous work. Kleindessner et al. [12] studied the *spectral clustering with fairness constraints*, where the fairness is defined as the prevention of over presentation. They restrict an upper bound of selected centers in every color for each cluster, which in the uniform case is essentially the upper bound version of the DB-MC-Center problem, which we call the *multi-colored center* (MC-Center) problem in this paper. As the main contribution, they presented a  $(3 \cdot 2^{\omega-1} - 1)$ -approximation algorithm running in  $O(nk\omega^2 + k\omega^4)$  time. Chen et al. [4] considered a generalization of the MC-Center problem called the *matroid median* problem, and gave a 3-approximation algorithm. Jones et al. [11] designed a 3-approximation algorithm and improved the time complexity of [4] to  $O(nk)$  via a maximum matching embedding technique. Note that a natural lower bound for this problem is  $2 - \varepsilon$ . We denote the *red-blue center* (RB-Center) problem as the two-colored case of the MC-Center problem.

The  $k$ -*median* problem is another well-known clustering problem. In the  $k$ -*median* problem, given are a set of points  $V$  in a metric space and an integer  $k$ . The objective is to find a  $k$ -sized subset such that the sum of distances between a point to its nearest center is minimized. Charikar et al. [3] gave the first constant approximation algorithm based on the filtering technique proposed in [14]. Using dependent-rounding, Byrka et al. [2] presented the currently best 2.675-approximation algorithm. The notion of diversity bounds had also be proposed in the  $k$ -*median* problem, but only one side bound requirement was considered. We refer to [6, 8, 13, 15, 17] for the upper bound version and [16] for the lower bound version.

Many other fairness measures have been proposed and investigated. For example, Bandyapadhyay et al. [1] considered a so-called *colorful  $k$ -center* clustering model from the clients' perspective, which requests for each color a coverage constraint. That means, any feasible clustering scheme should cover at least  $r_i$  points in color  $i$  for the fairness purpose. Based on a beautiful sparse linear programming technique, they presented a pseudo approximation algorithm for

the general metric space which violates the cardinality constraint, but can be refined to a true approximation with constant performance guarantee when the given point set is located in the plane. This result was improved to an efficient 3-approximation algorithm (2-approximation for a well separated special case) by Jia et al. [10] that either opens more than  $k$  centers or only works when the input points are in the plane. They built an interesting connection between the fair  $k$ -center with the well-known subset sum to illustrate the impossibility of achieving a good approximation through linear programming technique and overcome this by giving a nearly tight approximation guarantee. Very recently, Han et al. [9] considers the individually fair  $k$ -center with outliers, which provides the possibility of combination of different fairness measures in clustering.

As our main contribution, we study two diversity-bounded center problems, i.e., the DB-RB-Center problem and the DB-MC-Center problem. For the DB-RB-Center problem, we propose a 3-approximation algorithm. Inspiring by the work of [16], the main idea of our algorithm is based on guessing the numbers of selected red and blue centers in an optimal solution of the given DB-RB-Center instance. After the guessing process, we only need to deal with an RB-Center instance. It is worth mentioning that the algorithms in [4, 11] can directly be used to solve the RB-Center instance. For the sake of completeness, we give a simple approximation algorithm for the RB-Center problem. Our algorithm could be viewed as a reinterpretation of the algorithm in [4]. The main difference between these two algorithms is that our algorithm considers to construct and solve an matching problem in stead of a matroid intersection problem. For the DB-MC-Center problem, we extend the algorithm for its two-colored case, and give it a fixed parameterized tractable approximation algorithm with an approximation ratio of 3.

The remainder of this paper is structured as follows. We propose the approximation algorithms for the DB-RB-Center and DB-MC-Center problems in Sects. 2 and 3, respectively. Section 4 gives some discussions.

## 2 The Diversity-Bounded Red-Blue Center Problem

In this section, we propose a constant approximation algorithm for the DB-RB-Center problem. We first give the problem description of the DB-RB-Center problem in Subsect. 2.1. Then, we give an approximation algorithm for the RB-Center problem in Subsect. 2.2, which is used as a very important subroutine in our main algorithm for the DB-RB-Center problem. Finally, in Subsect. 2.3, the main algorithm is provided.

### 2.1 Problem Description of the DB-RB-Center Problem

In a DB-RB-Center instance  $\mathcal{I}_{DR}$ , we are given an integer  $k$  and a point set  $V$ , being composed by two disjoint point sets  $V_r$  and  $V_b$ . Each point in set  $V_r$  (resp.  $V_b$ ) is called a red (resp., blue) point. Integral lower bounds  $l_r$  and  $l_b$ , and integral upper bounds  $u_r$  and  $u_b$  are also given. Each pair of points  $(i, j)$ , where



$i, j \in V$ , is associated a distance  $d_{ij}$ . Assume that these distances are non-negative, symmetric, and satisfy the triangle inequality. The goal is to select some points  $O \subseteq V$  as centers, assign each point  $i \in V$  to some selected center  $\sigma(i) \in O$ , such that

- $l_r \leq |O \cap V_r| \leq u_r$ , and  $l_b \leq |O \cap V_b| \leq u_b$ ,
- $|O \cap V| = |O \cap V_r| + |O \cap V_b| \leq k$ ,
- the maximum distance of  $d_{\sigma(i)i}$  of a point  $i$  in  $V$  is minimized.

To ensure the DB-RB-Center problem has feasible solutions, we assume that  $l_r + l_b \leq k$  for any instance. Let  $(O, \sigma)$  denote a solution of a DB-RB-Center instance, in which  $O$  is the set of selected centers and  $\sigma : V \rightarrow O$  is a mapping from each point to its assigned center. For a solution  $(O, \sigma)$ , denote by  $C(O, \sigma)$  the maximum distance of a point in  $V$  to its assigned center in  $O$ , i.e.,

$$C(O, \sigma) = \max_{i \in V} d_{\sigma(i)i}.$$

Denote by  $(O_{\text{DR}}^*, \sigma_{\text{DR}}^*)$  an optimal solution of a DB-RB-Center instance, and denote by  $\text{OPT}_{\text{DR}}$  its objective value. Therefore,

$$\text{OPT}_{\text{DR}} = C(O_{\text{DR}}^*, \sigma_{\text{DR}}^*) = \max_{i \in V} d_{\sigma_{\text{DR}}^*(i)i}.$$

## 2.2 Algorithm for the RB-Center Problem

If  $l_r = 0, l_b = 0$ , and  $k = u_r + u_b$ , the DB-RB-Center problem simplifies to the RB-Center problem. We still use  $(O, \sigma)$  to denote a solution of an RB-Center instance, and use  $C(O, \sigma)$  to denote its objective value. Denote by  $(O_{\text{R}}^*, \sigma_{\text{R}}^*)$  an optimal solution of an RB-Center instance, and denote by  $\text{OPT}_{\text{R}}$  its objective value. Therefore,

$$\text{OPT}_{\text{R}} = C(O_{\text{R}}^*, \sigma_{\text{R}}^*) = \max_{i \in V} d_{\sigma_{\text{R}}^*(i)i}.$$

Now, we present an approximation algorithm for the RB-Center problem involving the optimal objective value  $\text{OPT}_{\text{R}}$ . We could suppose that  $\text{OPT}_{\text{R}}$  is known, by running the algorithm on each distance between a pair of points and taking the solution with the minimum objective value. The main algorithm consists of two essential phases. The first phase defines a partition of the point set  $V$ , and the second phase selects suitable centers.

### Phase 1: Define a Partition

For each point  $i \in V$ , let  $P(i)$  and  $N(i)$  be all the points within a distance of  $\text{OPT}_{\text{R}}$  and  $2\text{OPT}_{\text{R}}$  from  $i$ , respectively. Phase 1 arbitrarily chooses a point  $i \in V$  at the beginning, and then continues to find points that are not too close to all the already chosen points until no such points can be found. All the chosen points help to form a partition of  $V$ . The formal description of Phase 1 is given in Algorithm 1.

---

**Algorithm 1.** Algorithm for Defining a Partition of  $V$

---

**Input:** The point set  $V$  and distances  $\{d_{ij}\}_{i,j \in V}$  of an RB-Center instance  $\mathcal{I}_R$ .

**Output:** A partition of the point set  $V$ .

- 1 Initially, set  $C := \emptyset$  and  $U := V$ . For each point  $i \in V$ , set  $P(i) := \{j \in V : d_{ij} \leq \text{OPT}_R\}$ , and set  $N(i) := \{j \in V : d_{ij} \leq 2\text{OPT}_R\}$ .
  - 2 **While**  $U \neq \emptyset$  **do**  
 Arbitrarily choose a point  $i \in U$ . Update  $C := C \cup \{i\}$  and  $U := U \setminus N(i)$ .
  - 3 Define a partition of  $V$  as  $\{\{P(i)\}_{i \in C}, V \setminus \bigcup_{i \in C} P(i)\}$ , and output the partition.
- 

Note that Algorithm 1 does give a real partition of the point set  $V$ , since for any two distinct points  $i, j \in C$ , we always have that  $P(i) \cap P(j) = \emptyset$ . If there is a point  $q$  belongs to both  $P(i)$  and  $P(j)$ , then  $d_{ij} \leq d_{iq} + d_{qj} \leq 2\text{OPT}_R$ , contradicting the fact that the distance between  $i$  and  $j$  must be greater than  $2\text{OPT}_R$ .

**Phase 2: Select Centers**

Phase 2 selects a set of centers  $O \subseteq V$  for the RB-Center instance. Based on the partition obtained from Algorithm 1, Phase 2 first constructs a bipartite graph, and then finds its maximum matching, which implies the set of selected centers. The formal description of Phase 2 is given in Algorithm 2.

Towards better understanding, Fig. 1 provides an illustration of Phase 2.

It can be seen that the set of centers  $O$  is a feasible center set for the given RB-Center instance, since all the chosen edges in the maximum matching cannot incident to more than  $u_r$  (resp.,  $u_b$ ) shrunken points contracted from  $V_r$  (resp.,  $V_b$ ).

**The Main Algorithm**

Combining Algorithms 1 and 2 yields the main algorithm for the RB-Center problem, which is presented in Algorithm 3.

The following theorem gives the approximation ratio of Algorithm 3 for the RB-Center problem.

**Theorem 1.** *Algorithm 3 is a 3-approximation algorithm for the RB-Center problem.*

*Proof.* Recall that Phase 2 ensures that the solution  $(O, \sigma)$  obtained from Algorithm 3 is a feasible solution. With the following claim, we can prove the approximation ratio of Algorithm 3.

**Claim.** *The number of edges of a maximum matching of the bipartite graph  $G = (V_1, V_2, E')$  is exactly  $|C|$ .*

The claim implies that each  $P(i)$ , where  $i \in C$ , contains a point being selected as a center in  $O'$  ( $\subseteq O$ ). For any point  $i \in V$ , denote by  $i_{de}$  the point in  $C$

---

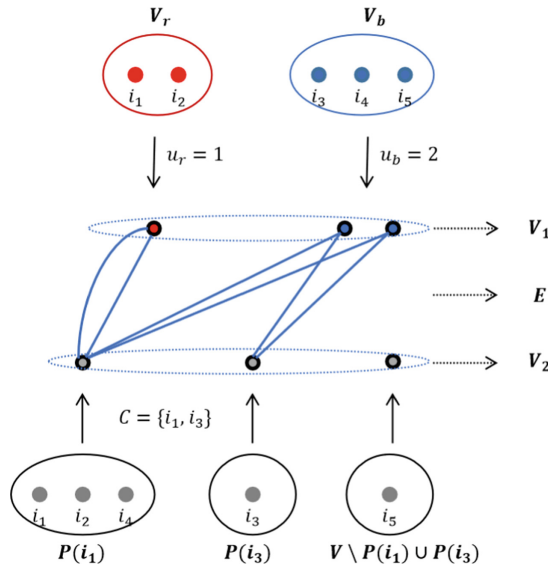
**Algorithm 2.** Algorithm for Selecting Centers

---

**Input:** The point sets  $V_r$  and  $V_b$ , upper bounds  $u_r$  and  $u_b$  of an RB-Center instance  $\mathcal{I}_R$ , and the partition  $\{\{P(i)\}_{i \in C}, V \setminus \bigcup_{i \in C} P(i)\}$  obtained from Algorithm 1.

**Output:** A set of the centers  $O \subseteq V$ .

- 1 Contract the point sets  $V_r$  and  $V_b$  to shrunken points named  $i_r$  and  $i_b$ , respectively. Copy the points  $i_r$  for  $u_r$  times and  $i_b$  for  $u_b$  times. Let  $V_1$  be all the shrunken points constructed in this step.
  - 2 Contract each point set in the partition  $\{\{P(i)\}_{i \in C}, V \setminus \bigcup_{i \in C} P(i)\}$  to a shrunken point. Let  $V_2$  be all the shrunken points constructed in this step, and let  $i_{ex}$  be the shrunken point corresponding to  $V \setminus \bigcup_{i \in C} P(i)$ .
  - 3 For a shrunken point  $i \in V_1 \cup V_2$ , let  $V(i)$  be all the points being contracted to  $i$ . For a pair of shrunken points  $(i_1, i_2)$ , where  $i_1 \in V_1$  and  $i_2 \in V_2 \setminus \{i_{ex}\}$ , if there exists a point  $i \in V$  belongs to both  $V(i_1)$  and  $V(i_2)$ , construct an edge between  $i_1$  and  $i_2$ , and mark this edge with a label  $i$ . Let  $E$  be all the constructed edges.
  - 4 For a pair of shrunken points  $(i_1, i_2)$ , where  $i_1 \in V_1$  and  $i_2 \in V_2$ , if there are multiple edges in  $E$  between them, arbitrarily choose one of the edges and delete all the others to obtain an edge set  $E'$ . For each deleted edge, attach its label to the edge that caused it to be deleted.
  - 5 Define a bipartite graph as  $G = (V_1, V_2, E')$ , and find its maximum matching. For each edge of the matching, choose exactly one label of the edge. Let  $O'$  be the set of points corresponding to these chosen labels. If  $|O' \cap V_r| \leq u_r$  (resp.,  $|O' \cap V_b| \leq u_b$ ), select some points in  $V_r \setminus O'$  (resp.,  $V_b \setminus O'$ ) to be centers so that the number of red (resp., blue) centers is exactly  $u_r$  (resp.,  $u_b$ ). Add all the additionally selected centers into  $O'$  to obtain  $O$ , and output  $O$ .
- 



**Fig. 1.** An illustration of Algorithm 2.

---

**Algorithm 3.** Main Algorithm for the RB-Center problem

---

**Input:** An RB-Center instance  $\mathcal{I}_R$  with inputs of  $V, V_r, V_b, u_r, u_b$  and  $\{d_{ij}\}_{i,j \in V}$ .

**Output:** A solution  $(O, \sigma)$  of the RB-Center instance.

- 1 Run Algorithm 1 to obtain a partition of  $V$ .
  - 2 Based on the partition, run Algorithm 2 to obtain the set of selected centers  $O$ .
  - 3 For each  $i \in V$ , set  $\sigma(i) := \arg \min_{o \in O} d_{oi}$ .
  - 4 Output  $(O, \sigma)$ .
- 

caused  $i$  to be removed from the set  $U$  in Step 2 of Algorithm 1. We have that  $d_{i_{de}i} \leq 2OPT_R$ . Note that  $i_{de}$  belongs to  $C$  and is within a distance of  $OPT_R$  from some selected center  $i'$  in  $O'$ . Therefore,

$$d_{i'i} \leq d_{i'i_{de}} + d_{i_{de}i} \leq 3OPT_R,$$

implying that the distance from any point  $i \in V$  to its nearest selected center in  $O$  is no more than  $3OPT_R$ . This completes the proof.

**Proof of the Claim.** We could demonstrate that there exists a matching that has exactly  $|C|$  edges. For each  $i \in C$ , recall that  $\sigma_R^*(i)$  is its assigned center in the optimal solution  $(O_R^*, \sigma_R^*)$ . Note that for any two distinct points  $i, j \in C$ , we have that  $P(i) \cap P(j) = \emptyset$ . Define  $C^* := \{\sigma_R^*(i)\}_{i \in C}$ . It is clear that each point  $i^* \in C^*$  corresponding to a distinct shrunken point contracted from the partition  $\{\{P(i)\}_{i \in C}, V \setminus \bigcup_{i \in C} P(i)\}$  obtained from Algorithm 1. Since  $C^* \subseteq O^*$ , we must have that  $|C^* \cap V_r| \leq u_r$  and  $|C^* \cap V_b| \leq u_b$ . Therefore, there must exist a matching in the bipartite graph  $G = (V_1, V_2, E')$ , which has exactly  $|C^*|$  (i.e.,  $|C|$ ) edges and marked with all the points in  $C^*$ .  $\square$

### 2.3 Algorithm for the DB-RB-Center Problem

In this subsection, we propose a constant approximation algorithm for the DB-RB-Center problem. The main idea of the algorithm based on guessing how many centers in  $V_r$  and  $V_b$  are selected in an optimal solution. So that solving the DB-RB-Center problem can be reduced to solving the RB-Center problem. The algorithm for the DB-RB-Center problem is provided in Algorithm 4.

The following theorem gives the approximation ratio of Algorithm 4 for the DB-RB-Center problem.

**Theorem 2.** *Algorithm 4 is a 3-approximation algorithm for the DB-RB-Center problem.*

*Proof.* For a DB-RB-Center instance, if we know how many centers in  $V_r$  and  $V_b$  are selected in its optimal solution, to solve the DB-RB-Center instance is to solve an RB-Center instance. In the case of  $u_r + u_b \leq k$ , an optimal solution would selected  $u_r$  red centers and  $u_b$  blue centers. In the case of  $l_r + l_b = k$ , an optimal solution would selected  $l_r$  red centers and  $l_b$  blue centers. In the case of  $l_r + l_b < k < u_r + u_b$ , we could enumerate all the feasible possibilities of the

---

**Algorithm 4.** Algorithm for the DB-RB-Center problem

---

**Input:** A DB-RB-Center instance  $\mathcal{I}_{DR}$  with inputs of  $V, V_r, V_b, u_r, u_b, l_r, l_b, k$ , and  $\{d_{ij}\}_{i,j \in V}$ .

**Output:** A solution  $(O, \sigma)$  of the DB-RB-Center instance.

**1 If**  $u_r + u_b \leq k$  **then**

Construct an RB-Center instance with inputs of  $V, V_r, V_b, u_r, u_b$  and  $\{d_{ij}\}_{i,j \in V}$ . Run Algorithm 3 to solve this RB-Center instance and obtain a solution  $(O, \sigma)$ . Output  $(O, \sigma)$ .

**2 If**  $l_r + l_b = k$  **then**

Construct an RB-Center instance with inputs of  $V, V_r, V_b, u'_r, u'_b$  and  $\{d_{ij}\}_{i,j \in V}$ , where  $u'_r = l_r$  and  $u'_b = l_b$ . Run Algorithm 3 to solve this RB-Center instance and obtain a solution  $(O, \sigma)$ . Output  $(O, \sigma)$ .

**3 If**  $l_r + l_b < k < u_r + u_b$  **then**

Define  $RB := \{(r, b) : l_r \leq r \leq u_r, l_b \leq b \leq u_b, r + b \leq k, r \in \mathbb{Z}, b \in \mathbb{Z}\}$ . For each  $(r, b)$  in  $RB$ , construct an RB-Center instance with inputs of  $V, V_r, V_b, u'_r, u'_b$  and  $\{d_{ij}\}_{i,j \in V}$ , where  $u'_r = r$  and  $u'_b = b$ . Run Algorithm 3 to solve all the constructed RB-Center instances and find a solution  $(O, \sigma)$  with minimum objective value. Output  $(O, \sigma)$ .

---

number of selected red and blue centers in an optimal solution, and construct the corresponding RB-Center instances. It can be seen that there are at most  $O(k^2)$  such instances. Therefore, Algorithm 4 is a polynomial time algorithm with the same approximation ratio as Algorithm 3.  $\square$

### 3 The Diversity-Bounded Multi-colored Center Problem

In this section, we extend Algorithm 4 to solve the DB-MC-Center problem, in which the given point set  $V$  could be composed by multiple disjoint point sets.

#### 3.1 Problem Description of the DB-MC-Center Problem

In a DB-MC-Center instance  $\mathcal{I}_{DM}$ , we are given an integer  $k$  and a point set  $V$ , being composed by several disjoint point sets  $V_1, V_2, \dots, V_\omega$ . Each point set  $V_t$ , where  $t \in \{1, 2, \dots, \omega\}$ , is associated with an integral lower bound  $l_t$  and an integral upper bound  $u_t$ . Each pair of points  $(i, j)$ , where  $i, j \in V$ , is associated a distance  $d_{ij}$ . Assume that these distances are non-negative, symmetric, and satisfy the triangle inequality. The goal is to select some points  $O \subseteq V$  as centers, assign each point  $i \in V$  to some selected center  $\sigma(i) \in O$ , such that

- $l_t \leq |O \cap V_t| \leq u_t$  for any  $t \in \{1, 2, \dots, \omega\}$ ,
- $|O \cap V| = \sum_{t=1}^{\omega} |O \cap V_t| \leq k$ ,
- the maximum distance of  $d_{\sigma(i)i}$  of a point  $i$  in  $V$  is minimized.

To ensure the DB-MC-Center problem has feasible solutions, we assume that  $\sum_{t=1}^{\omega} l_t \leq k$  for any instance. We still use  $(O, \sigma)$  to denote a solution of a DB-MC-Center instance, and use  $C(O, \sigma)$  to denote its objective value. Denote by  $(O_{DM}^*, \sigma_{DM}^*)$  an optimal solution of a DB-MC-Center instance, and denote by  $OPT_{DM}$  its objective value. Therefore,

$$OPT_{DM} = C(O_{DM}^*, \sigma_{DM}^*) = \max_{i \in V} d_{\sigma_{DM}^*(i)}(i).$$

### 3.2 Algorithm for the MC-Center Problem

If  $l_t = 0$  for any  $t \in \{1, 2, \dots, \omega\}$  and  $k = \sum_{t=1}^{\omega} u_t$ , the DB-MC-Center problem simplifies to the MC-Center problem. We still use  $(O, \sigma)$  to denote a solution of an MC-Center instance, and use  $C(O, \sigma)$  to denote its objective value. Denote by  $(O_M^*, \sigma_M^*)$  an optimal solution of an MC-Center instance, and denote by  $OPT_M$  its objective value. Therefore,

$$OPT_M = C(O_M^*, \sigma_M^*) = \max_{i \in V} d_{\sigma_M^*(i)}(i).$$

We could also suppose that  $OPT_M$  is known, since it must be equal to some distance between a pair of points in  $V$ . The algorithm for the RB-Center problem can be adapted to an algorithm for the MC-Center problem, which also consists of two phases. The first phase aims to define a partition of the point set  $V$  and is exactly same as Algorithm 1 except that we use  $OPT_M$  instead of  $OPT_R$ . The second phase is slightly different from Algorithm 2 and is given in Algorithm 5.

---

#### Algorithm 5. Algorithm for Selecting Centers

---

**Input:** The point sets  $\{V_t\}_{t \in \{1, 2, \dots, \omega\}}$ , upper bounds  $\{u_t\}_{t \in \{1, 2, \dots, \omega\}}$  of an MC-Center instance  $\mathcal{I}_M$ , and the partition  $\{\{P(i)\}_{i \in C}, V \setminus \bigcup_{i \in C} P(i)\}$  obtained from Algorithm 1.

**Output:** A set of the centers  $O \subseteq V$ .

- 1 For each  $t \in \{1, 2, \dots, \omega\}$ , contract each point set  $V_t$  to a shrunken point named  $i_t$ . Copy each point  $i_t$  for  $u_t$  times. Let  $V_1$  be all the shrunken points constructed in this step.
  - 2 Same as Step 2 in Algorithm 2.
  - 3 Same as Step 3 in Algorithm 2
  - 4 Same as Step 4 in Algorithm 2.
  - 5 Define a bipartite graph as  $G = (V_1, V_2, E')$ , and find its maximum matching. For each edge of the matching, choose exactly one label of the edge. Let  $O'$  be the set of points corresponding to these chosen labels. If  $|O' \cap V_t| \leq u_t$  for some  $t \in \{1, 2, \dots, \omega\}$ , select some points in  $V_t \setminus O'$  to be centers so that the number of centers in  $V_t$  is exactly  $u_t$ . Add all the additionally selected centers into  $O'$  to obtain  $O$ , and output  $O$ .
- 

Combining Algorithms 1 and 5 yields the algorithm for the MC-Center problem, which is presented in Algorithm 6. The following theorem gives the approximation ratio of Algorithm 6 for the MC-Center problem.

---

**Algorithm 6.** Main Algorithm for the MC-Center problem

---

**Input:** An MC-Center instance  $\mathcal{I}_M$  with inputs of  $V$ ,  $\{V_t\}_{t \in \{1,2,\dots,\omega\}}$ ,  $\{u_t\}_{t \in \{1,2,\dots,\omega\}}$ , and  $\{d_{ij}\}_{i,j \in V}$ .

**Output:** A solution  $(O, \sigma)$  of the RB-Center instance.

- 1 Run Algorithm 1 to obtain a partition of  $V$ .
  - 2 Based on the partition, run Algorithm 5 to obtain the set of selected centers  $O$ .
  - 3 For each  $i \in V$ , set  $\sigma(i) := \arg \min_{o \in O} d_{oi}$ .
  - 4 Output  $(O, \sigma)$ .
- 

**Theorem 3.** *Algorithm 6 is a 3-approximation algorithm for the MC-Center problem.*

Since the proof of this theorem is similar to the one for Theorem 1, we omit the proof here.

### 3.3 Algorithm for the DB-MC-Center Problem

The algorithm for the DB-MC-Center problem is given in Algorithm 7.

---

**Algorithm 7.** Algorithm for the DB-MC-Center problem

---

**Input:** A DB-MC-Center instance  $\mathcal{I}_{DR}$  with inputs of  $V$ ,  $\{V_t\}_{t \in \{1,2,\dots,\omega\}}$ ,  $\{u_t\}_{t \in \{1,2,\dots,\omega\}}$ ,  $\{l_t\}_{t \in \{1,2,\dots,\omega\}}$ ,  $k$ , and  $\{d_{ij}\}_{i,j \in V}$ .

**Output:** A solution  $(O, \sigma)$  of the DB-MB-Center instance.

- 1 **If**  $\sum_{t=1}^{\omega} u_t \leq k$  **then**  
 Construct an MC-Center instance with inputs of  $V$ ,  $\{V_t\}_{t \in \{1,2,\dots,\omega\}}$ ,  $\{u_t\}_{t \in \{1,2,\dots,\omega\}}$ , and  $\{d_{ij}\}_{i,j \in V}$ . Run Algorithm 6 to solve this MC-Center instance and obtain a solution  $(O, \sigma)$ . Output  $(O, \sigma)$ .
  - 2 **If**  $\sum_{t=1}^{\omega} l_t = k$  **then**  
 Construct an MC-Center instance with inputs of  $V$ ,  $\{V_t\}_{t \in \{1,2,\dots,\omega\}}$ ,  $\{u'_t\}_{t \in \{1,2,\dots,\omega\}}$ , and  $\{d_{ij}\}_{i,j \in V}$ , where  $u'_t = l_t$  for each  $t \in \{1, 2, \dots, \omega\}$ . Run Algorithm 6 to solve this MC-Center instance and obtain a solution  $(O, \sigma)$ . Output  $(O, \sigma)$ .
  - 3 **If**  $\sum_{t=1}^{\omega} l_t < k < \sum_{t=1}^{\omega} u_t$  **then**  
 Define  $MC := \{(r_1, r_2, \dots, r_{\omega}) : l_t \leq r_t \leq u_t \text{ for any } t \in \{1, 2, \dots, \omega\}, \sum_{t=1}^{\omega} r_t \leq k, r_t \in \mathbb{Z} \text{ for any } t \in \{1, 2, \dots, \omega\}\}$ . For each  $(r_1, r_2, \dots, r_{\omega})$  in  $MC$ , construct an MC-Center instance with inputs of  $V$ ,  $\{V_t\}_{t \in \{1,2,\dots,\omega\}}$ ,  $\{u'_t\}_{t \in \{1,2,\dots,\omega\}}$ , and  $\{d_{ij}\}_{i,j \in V}$ , where  $u'_t = r_t$  for any  $t \in \{1, 2, \dots, \omega\}$ . Run Algorithm 6 to solve all the constructed MC-Center instances and find a solution  $(O, \sigma)$  with minimum objective value. Output  $(O, \sigma)$ .
- 

The following theorem gives the main result of Algorithm 7.

**Theorem 4.** *Algorithm 7 is a fixed parameterized tractable approximation algorithm for the DB-MC-Center problem with an approximation ratio of 3.*

*Proof.* For a DB-MC-Center instance, if we know how many centers in each point set  $V_t$ , where  $t \in \{1, 2, \dots, \omega\}$ , is selected in its optimal solution, to solve the DB-MC-Center instance is to solve an MC-Center instance. In the case of  $\sum_{t=1}^{\omega} u_t \leq k$ , an optimal solution would selected  $u_r$  centers for each  $V_t$ . In the case of  $\sum_{t=1}^{\omega} l_t = k$ , an optimal solution would selected  $l_r$  centers for each  $V_t$ . In the case of  $\sum_{t=1}^{\omega} l_t < k < \sum_{t=1}^{\omega} u_t$ , we could enumerate all the feasible possibilities of the number of selected centers of each color in an optimal solution, and construct the corresponding MC-Center instances. It can be seen that there is at most  $O(k^{|\omega|})$  such instances. Therefore, Algorithm 7 is a fixed parameterized tractable approximation algorithm with the same approximation ratio as Algorithm 6.

## 4 Discussions

In the future, what we are most interested in is how to extend our algorithms to solve other clustering problems considering diversity bounds from both sides.

## References

1. Bandyapadhyay, S., Inamdar, T., Pai, S., Varadarajan, K.R.: A constant approximation for colorful  $k$ -center. In: Proceedings of the Annual European Symposium on Algorithms (ESA), pp. 1–14 (2019)
2. 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), 1–31 (2017)
3. Charikar, M., Guha, S., Tardos, É., Shmoys, D.B.: A constant-factor approximation algorithm for the  $k$ -median problem. *J. Comput. Syst. Sci.* **65**(1), 129–149 (2002)
4. Chen, D.Z., Li, J., Liang, H., Wang, H.: Matroid and knapsack center problems. *Algorithmica* **75**(1), 27–52 (2016)
5. Chierichetti, F., Kumar, R., Lattanzi, S., Vassilvitskii, S.: Fair clustering through fairlets. In: Proceedings of the Neural Information Processing Systems, pp. 5029–5037 (2017)
6. Friggstad, Z., Zhang, Y.: Tight analysis of a multiple-swap heuristic for budgeted red-blue median. In: Proceedings of the International Colloquium on Automata, Languages, and Programming, p. 75 (2016)
7. Gonzalez, T.F.: Clustering to minimize the maximum intercluster distance. *Theoret. Comput. Sci.* **38**, 293–306 (1985)
8. Hajiaghayi, M.T., Khandekar, R., Kortsarz, G.: Budgeted red-blue median and its generalizations. In: Proceedings of the European Symposium on Algorithms, pp. 314–325 (2010)
9. Han, L., Xu, D., Xu, Y., Yang, P.: Approximation algorithms for the individually fair  $k$ -center with outliers. *J. Global Optim.* 1–16 (2022)
10. Jia, X., Sheth, K., Svensson, O.: Fair colorful  $k$ -center clustering. *Math. Program.* **192**(1), 339–360 (2022)
11. Jones, M., Nguyen, H., Nguyen, T.: Fair  $k$ -centers via maximum matching. In: Proceedings of the International Conference on Machine Learning, pp. 4940–4949 (2020)



12. Kleindessner, M., Samadi, S., Awasthi P., Morgenstern J.: Guarantees for spectral clustering with fairness constraints. In: Proceedings of the International Conference on Machine Learning, pp. 3458–3467 (2019)
13. Krishnaswamy, R., Kumar, A., Nagarajan, V., Sabharwal, Y., Saha, B.: The matroid median problem. In: Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1117–1130 (2011)
14. Lin, J.H., Vitter, J.S.:  $\epsilon$ -Approximations with minimum packing constraint violation. In: Proceedings of the Annual ACM Symposium on Theory of Computing, pp. 771–782 (1992)
15. Swamy, C.: Improved approximation algorithms for matroid and knapsack median problems and applications. *ACM Trans. Algorithms* **12**(4), 1–22 (2016)
16. Thejaswi, S., Ordozgoiti, B., Gionis, A.: Diversity-aware  $k$ -median: Clustering with fair center representation. In: Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pp. 765–780 (2021)
17. Xu, Y., Möhring, R.H., Xu, D., Zhang, Y., Zou, Y.: A constant FPT approximation algorithm for hard-capacitated  $k$ -means. *Optim. Eng.* **21**(3), 709–722 (2020)

## Author Index

- Amano, Yuki 237
- Bai, Mingfei 359  
Bai, Tian 249  
Bazhenov, Nikolay 79  
Benoist, Emile 189
- Chang, Hong 272  
Chau, Vincent 21  
Chen, Rong 128
- Dai, Sijia 322  
Du, Donglei 140  
Du, Liman 116
- Feng, Junkai 32  
Fertin, Guillaume 189  
Fu, Chenchen 21
- Gao, Guichen 322  
Gao, Suixiang 116  
Gasarch, William 155  
Ge, Steven 55  
Gorain, Barun 346  
Guo, Tiande 218  
Guo, Xinru 322
- Han, Congying 218  
Han, Lu 402  
Hu, Jia 218  
Hu, Shan 305  
Huang, Xiaohui 93
- Itoh, Toshiya 43, 55
- Jean, Géraldine 189  
Jena, Sangram K. 103  
Jiang, Yanjun 11  
Jin, Jing 140
- Kalisz, Vít 176  
Klavík, Pavel 176
- Lafourcade, Pascal 201  
Lamprou, Ioannis 281
- Laskowski, Michael 155  
Li, Min 1  
Li, Ping 272  
Li, Weidong 262  
Liu, Pengcheng 93  
Liu, Shuilian 402  
Liu, Xiaofei 262  
Liu, Yuezhu 1  
Liu, Zhicheng 140, 272  
Lyu, Yan 21
- Makino, Kazuhisa 237  
Miao, Cuixia 370  
Miyahara, Daiki 201  
Mizuki, Takaaki 201  
Mondal, Kaushik 346  
Mustafa, Manat 79
- Nandakumar, Satyadev 334  
Nurakunov, Anvar 79
- Pandit, Supantha 346  
Pulari, Subin 334
- Ran, Yingli 93  
Robert, Léo 201  
Ruangwises, Suthee 43
- S, Akhil 334  
Sheng, Zimo 305  
Sigalas, Ioannis 281  
Subramani, K. 103, 293  
Sun, Yunjing 1
- Tan, Xuehou 128
- Vaxevanakis, Ioannis 281  
Vishnoi, Prateek 382
- Wang, Longchun 370  
Wang, Qiyue 359  
Wang, Yijing 11, 393  
Wang, Yiwei 359  
Wojciechowski, Piotr 293  
Wu, Weiwei 21

Xiao, Mingyu 249  
Xu, Chunming 359  
Xu, Yicheng 402

Yan, Binghao 67  
Yang, Ruiqi 11, 32  
Yang, Wenguo 116  
Ye, Weina 11

Zeman, Peter 176  
Zhang, Dongmei 393  
Zhang, Peng 67

Zhang, Xiaoyan 140, 272  
Zhang, Yapu 32, 393  
Zhang, Yizheng 21  
Zhang, Yong 322, 402  
Zhang, Zhao 93  
Zhang, Zhenning 32, 393  
Zhao, Xueyang 67  
Zhao, Yuhan 370  
Zhou, Yi 305  
Zhu, Shaopeng 155  
Zissimopoulos, Vassilis 281  
Zou, Juan 370