Kazushi Ikeda · Yoshiumi Kawamura · Kazuhisa Makino · Satoshi Tsujimoto · Nobuo Yamashita · Shintaro Yoshizawa · Hanna Sumita *Editors*

Advanced Mathematical Science for Mobility Society

Springer

Advanced Mathematical Science for Mobility Society

Kazushi Ikeda · Yoshiumi Kawamura ·
Kazuhisa Makino · Satoshi Tsujimoto ·
Nobuo Yamashita · Shintaro Yoshizawa ·
Hanna Sumita
Editors

Advanced Mathematical Science for Mobility Society

Springer

Editors
Kazushi Ikeda
Graduate School of Science
and Technology
Nara Institute of Science and Technology
Ikoma, Nara, Japan

Kazuhisa Makino
Research Institute for Mathematical Science
Kyoto University
Kyoto, Kyoto, Japan

Nobuo Yamashita
Graduate School of Informatics
Kyoto University
Kyoto, Kyoto, Japan

Hanna Sumita
School of Computing
Tokyo Institute of Technology
Meguro, Tokyo, Japan

Yoshiumi Kawamura
R-Frontier Division
Toyota Motor Corporation (Japan)
Susono, Shizuoka, Japan

Satoshi Tsujimoto
Graduate School of Informatics
Kyoto University
Kyoto, Kyoto, Japan

Shintaro Yoshizawa
R-Frontier Division
Toyota Motor Corporation (Japan)
Toyota, Aichi, Japan

ISBN 978-981-99-9771-8 ISBN 978-981-99-9772-5 (eBook)
https://doi.org/10.1007/978-981-99-9772-5

This Springer imprint is published by the registered company Springer Nature Singapore Pte Ltd.
The registered company address is: 152 Beach Road, #21-01/04 Gateway East, Singapore 189721, Singapore

Paper in this product is recyclable.

Preface

The development of information and communication infrastructures coupled with data science and artificial intelligence has led to major changes, not only in industrial structures but also in social systems. This movement was promoted in accordance with the Fifth Science and Technology Basic Plan in Japan and is continuing with the current 6th Science and Technology Basic Plan. In particular, technological innovation is progressing in new areas, such as "Connected, Autonomous/Automated, Shared, and Electric" (CASE) in the automotive industry, and the concept of the car itself, which supports both society and industry, is beginning to undergo major changes. In an immediate response to this trend, Toyota Motor Corporation has been working to realize future mobility society and has advocated becoming a future mobility company by departing from its previous role as an automobile company.

Mathematical science supports the fundamental technologies of all industries and societies. Given its versatility, once a mathematical innovation occurs, its impact is far-reaching. The importance of mathematical science is expected to increase in the field of mobility; for example, blockchain and mechanism design are expected to become fundamental technologies for supporting sharing economy businesses in the future, however, basic research for shared mobility services has not progressed. In autonomous driving system, the automated movement of a group of cars is expected to become an issue, which extends beyond the automated movement of individual cars. Hence, mobility society requires the development of mathematical science.

Mathematical science is inherently open-minded. Issues in mathematical science related to mobility are not always obvious in the beginning; research must start with a bird's-eye view of related fields and identify the problems from there. Researchers must come together and integrate knowledge based on free ideas, rather than on experiments and observations. Mathematical science research on mobility cannot be restricted to a single researcher, laboratory, or academic society. Research projects that involve many diverse researchers, going beyond the framework of conventional academia-industry collaboration must be conceived.

Yoshizawa approached Nakamura, who was Dean of the Graduate School of Informatics at Kyoto University, with a proposal for a specific initiative, and, in the process, solicited support for the realization of the concept. This secured the cooperation of Kazuhisa Makino, a Professor at the Research Institute for Mathematical Sciences at Kyoto University, who specializes in research on mathematics related to society. The research has since been promoted as an activity that makes the most of the networks of the participating researchers. After a preparatory phase in 2019, the joint research project, *Advanced Mathematical Science for Mobility Society*, between Kyoto University and Toyota Motor Corporation started in April 2020. After Nakamura's retirement in March 2021, Nobuo Yamashita, a Professor at the Graduate School of Informatics, Kyoto University, took over the project.

In the mobility society of the future, automobiles will be required to cooperate with infrastructure and provide safe and secure systems with exchanging large amounts of data. Therefore, we extended a definition of mobility to the flow of people, things, and information. The goal of this book is to create an original research field of mathematical science that can transform these flows into a mobility society where people can live happily with smiles on their faces. This book is published on the recommendation of Noboru Kikuchi, Director of the Genesis Research Institute. He is a former Director of the Frontier Research Center of Toyota Motor Corporation. This publication summarizes the results of the "Research on Advanced Mathematical Science for Mobility Society" project over a 3-year period, from April 2020 to March 2023. We would be delighted if the results of these studies provide reliable ideas and methods for future research.

Our academia-industry collaboration has received a great deal of support, and, at the time of its launch in April 2020, involved over 50 researchers from more than 20 universities and research institutes across Japan, with Kyoto University as the research hub. We would like to express our deepest gratitude to those involved, as well as to the many administrative staff members who supported the operation of this research project.

March 2023

Yoshimasa Nakamura
Osaka Seikei University
Osaka, Japan

Emeritus of Kyoto University
Kyoto, Japan

Shintaro Yoshizawa
Toyota Motor Corporation (Japan)
Toyota, Aichi, Japan

Contents

Part I
Introduction

Chapter 1
Advanced Mathematical Science for Mobility Society

Yoshiumi Kawamura, Kazuhisa Makino, and Nobuo Yamashita

1.1 Current State of Mobility Society

Due to the rapid information processing by computers and the evolution of communication equipment, we have become a society in which a large amount of information is constantly generated, stored, and transmitted. Information is becoming more and more important, and society has come to rely heavily on it in our daily lives. The 5th Science and Technology Basic Plan in Japan, which has been implemented since 2016, aims to realize a future society so-called a *super smart society* (*Society 5.0*) [1] that creates social prosperity through the evolution of Information and Communication Technology (ICT) and the creation of new values and services. In 2018, the Ministry of Economy, Trade and Industry in Japan compiled the Guidelines for Promoting Digital Transformation (DX) [2]. In this guideline, DX means that companies (1) respond to drastic changes in the business environment, utilize data and digital technology, and create new products, services, and business models based on the needs of customers and society, and (2) establish competitive advantages by transforming not only their business processes but also their organizations and corporate cultures.

On the other hand, in the automotive industry, the concept of the car itself is about to change due to technological innovations called *CASE* and *MaaS*. CASE is a coined word that takes the initials of *Connected, Autonomous/Automated, Shared,*

Y. Kawamura (✉)
Toyota Motor Corporation, 1200, Mishuku, Susono, Shizuoka 410-1193, Japan
e-mail: yoshiumi_kawamura@mail.toyota.co.jp

K. Makino
Kyoto University, Kitashirakawa Oiwake-cho, Sakyo-ku, Kyoto 606-8502, Japan
e-mail: makino@kurims.kyoto-u.ac.jp

N. Yamashita
Kyoto University, Yoshida-honmachi, Sakyo-ku, Kyoto 606-8501, Japan
e-mail: nobuo@i.kyoto-u.ac.jp

K. Ikeda et al. (eds.), *Advanced Mathematical Science for Mobility Society*,
https://doi.org/10.1007/978-981-99-9772-5_1

and Electric, and represents the axis of future new vehicle development, while MaaS stands for *Mobility as a Service*, and indicates the future that transportation including cars, trains, and airplanes aims for. Automobiles that automatically run on electric power while exchanging large amounts of data over the Internet are nothing more than information terminals, and automobiles that are not owned but shared are already part of the social infrastructure.

In CASE, "Connected" means connection with the internet, and by connecting the car to the network in real time, it is possible to send and receive all kinds of data obtained while driving, such as map, accident, and weather information. "Autonomous/Automated" means self-driving without human intervention, and "Shared" means sharing the cars. It was common to purchase and use a car individually. Currently, it is popular to share a car so that it can be used whenever you want. Finally, "Electric" means environmentally friendly electric vehicles.

MaaS makes cloud-based and seamless transportation even if it consists of multiple operating units such as bus, train, and airplane. Even now, when traveling using multiple modes of transportation, you can use the route search to find out how to get to your destination and how long it will take, without having to follow the timetables one by one to derive the itinerary. Even so, it is still inefficient, since reserving and purchasing tickets must be made by each operator. In the world of MaaS, it is possible to search for the most suitable means of transportation, make reservations, and make payments all at once using smartphone apps.

By accelerating these technological innovations related to mobility, it is expected that significant contribution will be made to environmental issues that have become a challenge in recent years, particularly in reducing carbon emissions.

1.2 Project Titled Advanced Mathematical Science for Mobility Society

While the automotive industry has made steady progress in technological innovations under the names of CASE and MaaS, Toyota Frontier Research Center, one of Toyota Motor Corporation's research bases, considers that mathematical science is the most important tool for conceiving future mobility services, in light of Toyota's vision of turning mobility into social potential.

Mathematical technologies are, of course, adopted in many situations throughout the automotive industry. For example, mathematical optimization technique is widely used and contributes in some way, from design, manufacturing, to service design. In the future mobility society, however, it is not always possible to satisfy all requirements with today's mathematical technologies. For example, design space for the mobility service will become complex and large scale. Therefore, further progress will be required in optimization technology to find the optimal conditions in such complex spaces. It is also necessary to search for better solution and control the systems while responding to changing environments every moment, dynamically.

Furthermore, as individual values diversify in the future, we believe that it is necessary to design a mechanism that is fair to match each individual's value, not just the overall optimum, in order to provide a service in close contact with each customer.

Based on the above motivation, we started the joint project of Kyoto University and Toyota Motor Corporation, titled "Advanced Mathematical Science for Mobility Society" in 2020. The project is a collaboration between Toyota Frontier Research Center, Toyota Motor Corporation and Graduate School of Informatics and Research Institute for Mathematical Sciences (RIMS), Kyoto University. However, it involves not only researchers at Kyoto University and Toyota Motor Corporation, but also prominent domestic researchers in the fields of mathematical science and informatics. The project takes a broader view of the concepts of mobility and movement which includes the flow of people through public transportation, the distribution of information, and the energy flow to capture the essence of the future mobility society, and deals with the following three topics

1. Mathematical models of flow,
2. Mathematical methods for huge data and network analysis, and
3. Algorithms for mobility society.

The first topic is a mathematical model on flow. Here, we consider to handle the flow of objects (e.g., goods), people, and information. As for flow of objects and people, traffic flow is typical, but it also includes delivery of parts and products in the manufacturing process. The flow of "information" is also important for mobility society. For example, Toyota Motor Corporation uses a well-known "Toyota Production System" [3] technique for the production processes. Here, a flow diagram of objects and information, called "Value Stream Map", is created and the flow of information is visualized. This is because the stagnation of the information flow affects the stagnation of the production process. Thus, we have studied mathematical techniques to model the flow and to construct a smooth mobility society without stagnation.

The second topic then describes mathematical techniques dealing with enormous data on mobility from the viewpoints of machine learning, numerical analysis, statistical physics, etc. Needless to say, handling of big data is an important issue in the future. Especially with the progress of the connected technology, the type and size of the data to be handled become enormous. Even if the computational power advances, it will be difficult to effectively handle these enormous data without ingenuity. In order to compress necessary information effectively, or to extract useful information, the progress of basic mathematical technology is necessary. We will also consider techniques for handling sensitive data, such as personal privacy and company secret information, as well as mathematical techniques for secure distributed systems such as block-chains.

The last topic discusses algorithmic problems on mobile society. An example of a mobility system is a car-sharing service. Such a service requires, for example, both convenience for individuals and robustness and efficiency as a system. Mechanisms that are always optimal in response to changing situations every moment and the needs of people are also required. Furthermore, in the future society where the sense of value is diversifying, a mechanism that is fair to all customers will also be important.

From these viewpoints, this topic examines a method for constructing and analyzing algorithms for controlling and optimizing systems and services.

In this monograph, we discuss some of the results obtained by nine groups of the project on the three research topics above, as well as the related research areas.

The rest of the monograph is organized as follows. Part 2 discusses mathematical models of flow, which consists of two chapters.

Chapter 2:	Analysis of Autonomous Many-Body Particle Models from Geometric Perspective and its Applications
Chapter 3:	Integrable Systems Related to Matrix LR Transformations

and Part 3 discusses mathematical methods for huge data and network analysis:

Chapter 4:	Numerical Analysis for Data Relationship
Chapter 5:	Application of Tensor Network Formalism for Processing Tensor Data
Chapter 6:	Machine Learning Approach to Mobility Analyses
Chapter 7:	Graph Optimization Problems and Algorithms for DAG-type Blockchains

Finally, Part 4 treats algorithm issues for mobility society, consisting of the following three chapters.

Chapter 8:	System-Control-Based Approach to Car-Sharing Systems
Chapter 9:	Algorithms for Future Mobility Society
Chapter 10:	Mechanism Design for Mobility.

Acknowledgements We would like to thank Ms. Maiko Hori for her assistance in writing this chapter.

References

1. Cabinet Office, Government of Japan. https://www8.cao.go.jp/cstp/society5_0/index.html
2. Ministry of Economy, Trade and Industry, Japan. https://www.meti.go.jp/policy/it_policy/investment/dgc/dgc2.pdf
3. Toyota Motor Corporation. https://global.toyota/en/company/vision-and-philosophy/production-system/

Part II
Mathematical Models of Flow

Chapter 2
Analysis of Autonomous Many-Body Particle Models from Geometric Perspective and Its Applications

Satoshi Tsujimoto, Tsuyoshi Kato, Ryosuke Kojima, Kazuki Maeda, and Francesco Zanlungo

2.1 Introduction

Autonomous many-body particle systems, such as traffic flow models, pedestrian flow models, and molecular biology models, are very important targets that appear in various fields. Since the behavior of these systems is essentially nonlinear and is rich in variety and flexibility, it is important to grasp the whole picture of the system not only numerically, but also through theoretical analysis.

For interacting systems of many-body particles, we have been studying

A. the development of analytical methods,
B. the investigation and extension of fundamental models.

For A., we have mainly worked on the following analytical methods from the viewpoint of geometry: (i) derivation and analysis of the Burgers cellular automaton

S. Tsujimoto (✉)
Graduate School of Informatics, Kyoto University, Yoshida-honmachi, Sakyo-ku, Kyoto 606-8501, Japan
e-mail: tsujimoto.satoshi.5s@kyoto-u.ac.jp

T. Kato
Kyoto University, Kitashirakawa Oiwake-cho, Sakyo-ku, Kyoto 606-8502, Japan
e-mail: kato.tsuyoshi.5m@kyoto-u.ac.jp

R. Kojima
Kyoto University, 53 Kawahara-cho, Shogoin, Sakyo-ku, Kyoto 606-8507, Japan
e-mail: kojima.ryosuke.8e@kyoto-u.ac.jp

K. Maeda
The University of Fukuchiyama, 3370 Hori, Fukuchiyama, Kyoto 620-0886, Japan
e-mail: kmaeda@kmaeda.net

F. Zanlungo
International Professional University of Technology in Osaka, 3-3-1 Umeda, Kita-ku, Osaka 530-0001, Japan
e-mail: zanlungo@atr.jp

K. Ikeda et al. (eds.), *Advanced Mathematical Science for Mobility Society*,
https://doi.org/10.1007/978-981-99-9772-5_2

(BCA) and the box-ball system (BBS) by the methods of tropical geometry and ultra-discrete systems, (ii) various studies on many-body particle interaction systems based on discrete Morse theory from a phase-geometric approach, and (iii) development of a fundamental model for B., which is a model of many-body particle interaction systems. As primary models to be treated in B., there are various models such as traffic flow, pedestrian flow, and molecular biology models. In this paper, we will mainly focus on BCA, BBS, and the totally asymmetric simple exclusion process (TASEP), which will be briefly introduced in this paper. The method of tropical geometry is effective not only in analytical methods but also in the derivation of new models. In the modeling of particle systems, there are examples of successful extraction of the skeletal part in particular, and the BBS obtained from the Korteweg-de Vries (KdV) equation and its ultra-discretization of the nonlinear partial differential equation is well known. Although research results from this perspective are not yet at the level of basic theory, it is possible to investigate various extensions of cellular automata and analytical solutions to their initial value problems. As an unexpected byproduct of the above research, we were also able to clarify the relationship between the BBS and a computational procedure for invariant factors of integer matrices.

In Sect. 2.2, we outline the discrete Morse theory methods used. Section 2.3 gives an example of the application of the discrete Morse theory method to traffic flow model analysis. In Sect. 2.4, we derive a model with more general degrees of freedom by adding internal degrees of freedom, such as explicit and implicit degrees of freedom, to the particle system, and we also derive a quantum version of TASEP and discuss its results. Finally, in Sect. 2.5, we will discuss pedestrian flow models as the next subject and summarize the current status and future prospects.

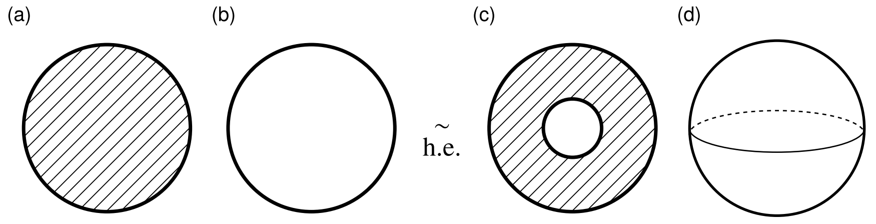
2.2 Discrete Morse Theory

In this section, we give a brief introduction to topology, especially homology, Morse theory, and discrete Morse theory.

2.2.1 Homology

The homology theory is a fundamental tool to study differentiable manifolds viewed as topological spaces. Homology groups $H_k(M)$, $k = 0, 1, 2, \ldots$, of a manifold M are abelian groups that represent topological invariants of M; i.e. two manifolds have the isomorphic homology groups if they can be continuously deformed into each other (homotopy equivalent; note that the reverse is not always true). In the simplest case, the kth homology group $H_k(M)$ becomes $\mathbb{Z}^n$ that means

- the case of $k = 0$: the number of connected components in M is n;
- the case of $k \geq 1$: roughly, the number of k-dimensional holes in M is n,



$H_0 = \mathbb{Z}$,	$H_0 = \mathbb{Z}$,	$H_0 = \mathbb{Z}$,	$H_0 = \mathbb{Z}$,
$H_k = 0$ for $k \geq 1$.	$H_1 = \mathbb{Z}$,	$H_1 = \mathbb{Z}$,	$H_1 = 0$,
	$H_k = 0$ for $k \geq 2$.	$H_k = 0$ for $k \geq 2$.	$H_2 = \mathbb{Z}$,
			$H_k = 0$ for $k \geq 3$.

Fig. 2.1 Examples of homology groups. **a** A disk has no hole. **b** A circle has a 1-dimensional hole. **c** An annulus also has a 1-dimensional hole. A circle and an annulus are homotopy equivalent. **d** A sphere has a 2-dimensional hole

where n is called the Betti number. Figure 2.1 gives several examples. In more complicated cases, the homology groups may have finite subgroups called torsion subgroups, but we do not explain them here.

If two manifolds have different homology, then they cannot be homotopy equivalent. Hence, the computation of the homology groups of differentiable manifolds is a first step to study topology of manifolds (see, e.g. [15]). There is Morse theory that is useful for the construction of homology groups.

2.2.2 *Morse Theory*

Let us consider two differentiable functions on a manifold $f, g\colon M \to \mathbb{R}$, where f and g do not have any degenerate critical points. The Morse theory is able to construct the isomorphic homology groups of M from both different functions f and g; i.e. any generic functions on a manifold M reflect the topology of M directly. A "simpler" function that has as few critical points as possible makes it easy to compute homology groups.

We introduce gradient vector fields here. Consider a situation that a local coordinate system with variables $x_1, x_2, \dots, x_n$ is given on $U \subset M$. A map $\varphi_U : U \to \mathbb{R}^n$ represents this coordinate. Then, the tangent space at a point $p \in U$ denoted by T_pM is defined as a linear space formally spanned by the differential operators $\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}, \dots, \frac{\partial}{\partial x_n}$. The tangent bundle of M is defined by $TM := \sqcup_{p\in M} T_pM$. For the tangent space T_pM, we can consider a dual space T_p^*M, which is called the cotangent space. The cotangent bundle $T^*M := \sqcup_{p\in M} T_p^*M$ is also defined. There is a basis $\{dx_1, dx_2, \dots, dx_n\}$ of T_p^*M satisfying

$$dx_i\left(\frac{\partial}{\partial x_j}\right) = \delta_{ij}, \quad i, j = 1, 2, \ldots, n,$$

where δ_{ij} is the Kronecker delta.

For a function $f\colon M \to \mathbb{R}$, the differential 1-form of f at $p \in U$ is defined by

$$df_p := \sum_{i=1}^{n} \frac{\partial(f \circ \varphi_U^{-1})}{\partial x_i} dx_i \in T_p^*M.$$

If a Riemannian metric $g_p \in T_p^*M \otimes T_p^*M$ is given at each point $p \in M$, then the gradient of f denoted by $\nabla f\colon M \to TM$; $p \mapsto \nabla f(p) \in T_pM$ is uniquely defined by the equation

$$g_p(\nabla f(p), v_p) = df_p(v_p)$$

for all $p \in M$ and $v_p \in T_pM$. Although this definition may look a bit complicated, this is exactly a generalization of the gradient on the Euclidean space. (Consider the case $g_p = \sum_{i=1}^{n} dx_i \otimes dx_i$ for all $p \in M$. That may be helpful for understanding the meaning.)

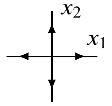
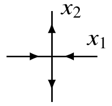
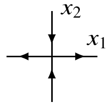
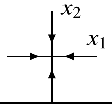
A point $\overline{p} \in M$ is called a critical point of f if $\nabla f(\overline{p}) = 0$. For the Hessian matrix of f at a point p defined by

$$H_f(p) := \begin{pmatrix} \frac{\partial^2(f\circ\varphi_U^{-1})}{\partial x_1^2} & \frac{\partial^2(f\circ\varphi_U^{-1})}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2(f\circ\varphi_U^{-1})}{\partial x_1 \partial x_n} \\ \frac{\partial^2(f\circ\varphi_U^{-1})}{\partial x_2 \partial x_1} & \frac{\partial^2(f\circ\varphi_U^{-1})}{\partial x_2^2} & \cdots & \frac{\partial^2(f\circ\varphi_U^{-1})}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2(f\circ\varphi_U^{-1})}{\partial x_n \partial x_1} & \frac{\partial^2(f\circ\varphi_U^{-1})}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2(f\circ\varphi_U^{-1})}{\partial x_n^2} \end{pmatrix} (\varphi_U(p)),$$

if $\det H_f(\overline{p}) \neq 0$ then $\overline{p}$ is called a non-degenerate critical point. If all the critical points of f are non-degenerate then f is called a Morse function. It is known that Morse functions are generic in the set of smooth functions. The index of a critical point $\overline{p}$ of f, denoted by λ in this section, is defined as the number of the negative eigenvalues of the Hessian matrix $H_f(\overline{p})$. Table 2.1 shows examples of critical points and their indices. We can see that critical points of index 0, 1, and 2 in the two-dimensional Euclidean space correspond to local minimum points, saddle points, and local maximum points, respectively. In the examples, we consider critical points on the Euclidean space, but we also can consider critical points on any manifolds in general. See Fig. 2.2.

From critical points and a gradient vector field, we can compute homology groups (called Morse homology) of M. Let p and q be critical points and $\mathcal{M}(p, q)$ be the set of all the integral curves of $-\nabla f$ from p to q; i.e. $\gamma \in \mathcal{M}(p, q)$ means $\frac{d\gamma}{dt}(t) = -\nabla f(\gamma(t))$, $\lim_{t\to-\infty} = p$ and $\lim_{t\to+\infty} = q$. For example, in Fig. 2.2, $\mathcal{M}(p_3, p_2)$ has two integral curves γ_1 and γ_2, and $\gamma_1 - \gamma_2$ makes a cycle. We give a

Table 2.1 Examples of critical points (the origin in these cases) and their indices λ on the two-dimensional Euclidean space

$f \circ \varphi_U^{-1}(x_1, x_2)$	∇f	gradient at the origin	λ
$x_1^2 + x_2^2$	$2x_1 \frac{\partial}{\partial x_1} + 2x_2 \frac{\partial}{\partial x_2}$		0
$-x_1^2 + x_2^2$	$-2x_1 \frac{\partial}{\partial x_1} + 2x_2 \frac{\partial}{\partial x_2}$		1
$x_1^2 - x_2^2$	$2x_1 \frac{\partial}{\partial x_1} - 2x_2 \frac{\partial}{\partial x_2}$		1
$-x_1^2 - x_2^2$	$-2x_1 \frac{\partial}{\partial x_1} - 2x_2 \frac{\partial}{\partial x_2}$		2

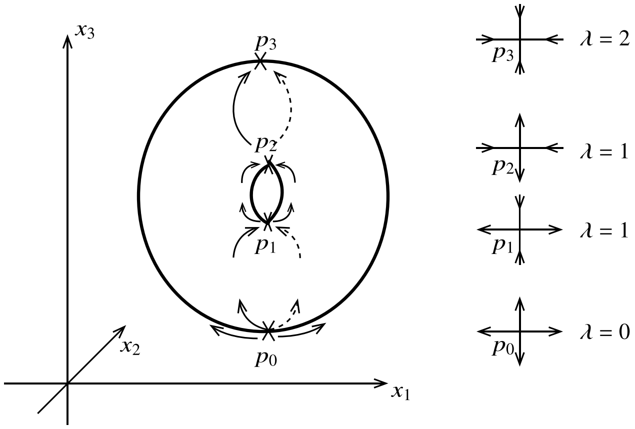


Fig. 2.2 An example of critical points on a two-dimensional manifold called a torus $T^2 = S^1 \times S^1$ embedded in the three-dimensional Euclidean space. The Morse function is defined by the height function $f(p) = x_3$

sign of integral curves $\epsilon\colon \mathcal{M}(p, q) \to \{\pm 1\}$ according to cycles if they exist. In the example, if we set $\epsilon(\gamma_1) = 1$, then another should have opposite sign $\epsilon(\gamma_2) = -1$.

Let us define the Morse complex of index λ by the free $\mathbb{Z}$-module generated by all the critical points of index λ

$$C_\lambda := \bigoplus_{\lambda_i = \lambda} \mathbb{Z} p_i,$$

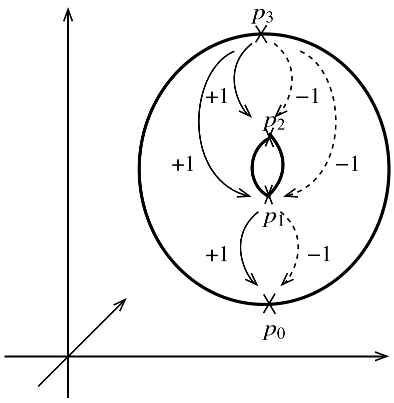


Fig. 2.3 Integral curves on the torus T^2 from critical points of index λ to ones of index $\lambda - 1$. There exist two curves from p_2 to p_0 that are not drawn

where λ_i denotes the index of the critical point p_i, and boundary operators $d_\lambda \colon C_\lambda \to C_{\lambda-1}$, which are homomorphisms, by

$$d_\lambda(p_i) := \sum_{\substack{\lambda_j=\lambda-1 \\ \gamma\in\mathcal{M}(p_i,p_j)}} \epsilon(\gamma)p_j \quad \text{for all critical points } p_i \in C_\lambda.$$

Since we can see that $d_{\lambda-1} \circ d_\lambda = 0$, $(C_\bullet, d_\bullet)$ is a chain complex. The kth Morse homology group is defined by

$$H_k(M) := \operatorname{Ker} d_k / \operatorname{Im} d_{k+1}, \quad k = 0, 1, 2, \dots .$$

For example, the Morse homology groups of the torus T^2 (Fig. 2.2) are computed as follows (see also Fig. 2.3): the Morse complexes are $C_0 = \mathbb{Z}p_0$, $C_1 = \mathbb{Z}p_1 \oplus \mathbb{Z}p_2$, $C_2 = \mathbb{Z}p_3$, and boundary operators are defined as $d_0(p_0) = 0$, $d_1(p_1) = (+1 - 1)p_0 = 0$, $d_1(p_2) = (+1 - 1)p_0 = 0$, $d_2(p_3) = (+1 - 1)p_1 + (+1 - 1)p_2 = 0$. Note that $C_k = 0$ for $k \geq 3$ and $k = -1$, and $d_k = 0$ for $k \geq 3$ and $k = 0$. Hence, $\operatorname{Ker} d_k = C_k$ and $\operatorname{Im} d_k = 0$ for all k, then we obtain $H_0(M) = C_0 \cong \mathbb{Z}$, $H_1(M) = C_1 \cong \mathbb{Z}^2$, and $H_2(M) = C_2 \cong \mathbb{Z}$. This example shows that the fewer critical points the Morse function has, the easier it is to compute the Morse homology.

One can find more details in, for example, Milnor's book [17].

2.2.3 Discrete Morse Theory

The discrete Morse theory is a combinatorial analog of the Morse theory developed by Forman [11, 12]. Although the discrete Morse theory is constructed on CW complexes abstractly in general, we explain the theory more concretely for simplicity.

We call the following k-dimensional cells, or simply k-cells:

- 0-cell: a point;
- 1-cell: a curve whose boundaries are points (0-cells);
- 2-cell: a surface whose boundaries are curves (1-cells);
- 3-cell: a solid whose boundaries are surfaces (2-cells);
- ...

A set of cells $\mathcal{K}$ is called a cell complex if a k-cell α is included in $\mathcal{K}$ implies that all the boundaries of α are included in $\mathcal{K}$. A cell α is called a face of β if $\alpha \subset \beta$, where we consider a cell as a set of vertices. A face α of β is called maximal if $\dim \alpha = \dim \beta - 1$.

A function $f : \mathcal{K} \to \mathbb{R}$ is called a discrete Morse function if the following conditions are satisfied: for all $\alpha \in \mathcal{K}$,

1. $\#\{\, \beta \in \mathcal{K} : \alpha \text{ is a maximal face of } \beta \text{ and } f(\alpha) \geq f(\beta) \,\} \leq 1$;
2. $\#\{\, \gamma \in \mathcal{K} :\ \gamma \text{ is a maxima face of } \alpha \text{ and } f(\gamma) \geq f(\alpha) \,\} \leq 1$.

A cell $\alpha \in \mathcal{K}$ is called a critical cell if the following strict conditions are satisfied:

1. $\#\{\, \beta \in \mathcal{K} : \alpha \text{ is a maximal face of } \beta \text{ and } f(\alpha) \geq f(\beta) \,\} = 0$;
2. $\#\{\, \gamma \in \mathcal{K} : \gamma \text{ is a maximal face of } \alpha \text{ and } f(\gamma) \geq f(\alpha) \,\} = 0$.

For a non-critical cell $\alpha \in \mathcal{K}$, there exists a cell $\beta \in \mathcal{K}$, where α is a maximal face of β and $f(\alpha) \geq f(\beta)$, or $\gamma \in \mathcal{K}$, where γ is a maximal face of α and $f(\gamma) \geq f(\alpha)$. For these pairs of cells, we draw arrows from α to β or from γ to α. The set of these arrows is called a gradient vector field. For example, see Fig. 2.4.

Given a gradient vector field of a discrete Morse function f on $\mathcal{K}$, we call a sequence of k-cells α_i and $(k+1)$-cells β_i alternately $(\alpha_1, \beta_1, \alpha_2, \beta_2, \ldots, \beta_r, \alpha_{r+1})$ a gradient path of f if, for each $i = 1, 2, \ldots, r$, a pair (α_i, β_i) is a vector of the gradient vector field, α_{i+1} is a face of β_i and $\alpha_{i+1} \neq \alpha_i$. By the definition, one can readily derive the relation $f(\alpha_1) \geq f(\beta_1) > f(\alpha_2) \geq f(\beta_2) > \cdots \geq f(\beta_r) > f(\alpha_{r+1})$. Then, we can obtain a way to compute the homology groups of $\mathcal{K}$ in a similar manner to the continuous Morse theory. Let C_λ be the Morse complex of dimension λ; i.e. C_λ

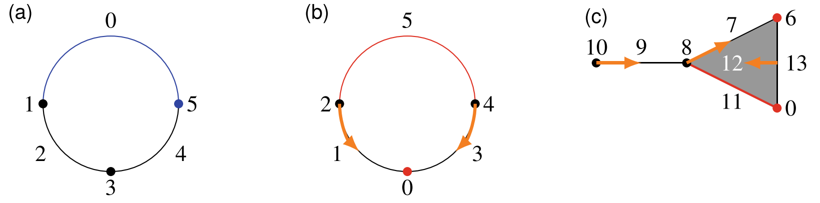


Fig. 2.4 Examples of cell complexes and discrete Morse functions. There are 0-cells (points), 1-cells (curves), and a 2-cell (surface; only in **c**), and numbers written next to each cell are the values of the discrete Morse functions. The function on **a** is not a discrete Morse function because blue cells do not satisfy the conditions. The functions on **b** and **c** are discrete Morse functions and red cells are critical cells. The orange arrows in **b** and **c** indicate a gradient vector fields

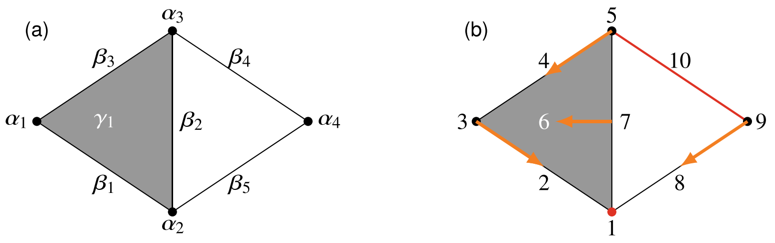


Fig. 2.5 An example of gradient paths and Morse homology groups. **a** indicates the names of the cells. **b** indicates the values of the discrete Morse function, the critical cells, and the gradient vector field

is the free $\mathbb{Z}$-module generated by all the critical λ-cells. Let $d_\lambda : C_\lambda \to C_{\lambda-1}$ be a boundary operator defined by

$$d_\lambda(\beta_i) := \sum_{\substack{\alpha_j:\,\text{critical }(\lambda-1)-cell \\ \gamma\in\mathcal{M}(\beta_i,\alpha_j)}} \epsilon(\gamma)\alpha_j \quad \text{for all critical cells } \beta_i \in C_\lambda,$$

where $\mathcal{M}(\beta_i, \alpha_j)$ is the set of all the gradient paths from a maximal face of β_i to the cell α_j, and $\epsilon(\gamma) \in \{\pm 1\}$ is determined depending on whether the chosen orientation on β_i induces the same orientation of the gradient path on α_j or the opposite orientation. Then, the Morse homology is defined by $H_k(\mathcal{K}) = \text{Ker}\, d_k/\text{Im}\, d_{k+1}$, $k = 0, 1, 2, \ldots$.

For example, consider the cell complex and the discrete Morse function in Fig. 2.5. There are a critical 0-cell α_2 and a critical 1-cell β_4, and gradient paths (α_4, β_5, α_2) whose sign is $+1$ and (α_3, β_3, α_1, β_1, α_2) whose sign is -1. There are the complexes $C_0 = \mathbb{Z}\alpha_2$, $C_1 = \mathbb{Z}\beta_4$, $C_2 = 0$, and the boundary operators $d_0(\alpha_2) = 0$, $d_1(\beta_4) = (+1 - 1)\alpha_2 = 0, d_2 = 0$. Hence, $\text{Ker}\, d_k = C_k$ and $\text{Im}\, d_k = 0$ for all k, then we obtain $H_0(\mathcal{K}) = C_0 \cong \mathbb{Z}$, $H_1(\mathcal{K}) = C_1 \cong \mathbb{Z}$, and $H_2(\mathcal{K}) = 0$.

2.3 Application of Discrete Morse Theory to Traffic Flow Models

In this section, we consider an application of the discrete Morse theory, which was introduced in Sect. 2.2, to a simple traffic flow model called the Burgers cellular automaton.

2.3.1 Algorithms for Constructing Discrete Morse Functions on Cubical Complexes

To apply the discrete Morse theory, we first have to construct a complex and a discrete Morse function that possesses topological information of the data to be analyzed. We adapt the algorithms for constructing discrete Morse functions on cubical complexes to analyze 2D and 3D grayscale digital images proposed by Robins, Wood, and Sheppard [21]. The following is a brief exposition of the algorithm for 2D lattice. The algorithm for 3D lattice can be also described in the same manner.

Let us consider a discrete lattice

$$D = \{\, (i, j) \in \mathbb{Z}^2 : 0 \le i \le I,\ 0 \le j \le J \,\}.$$

We can derive a cubical complex $\mathcal{K}$ from the lattice D as follows: the vertices $(i, j) \in D$ are 0-cells of the complex $\mathcal{K}$. The unit edges between the vertices, and the unit squares are 1-cells and 2-cells of the complex $\mathcal{K}$, respectively.

Let $g\colon D \to \mathbb{R}$ be a function that represents a given numerical data on the lattice D. We assume that the vertices in D can be strictly totally ordered as $g(x_0) < g(x_1) < \cdots < g(x_N)$. Note that this requirement is always satisfied by adding perturbation to g without giving adverse effects to the results of the algorithm. Then, subsets of the neighboring cells of $x \in D$, called the lower stars of x,

$$L(x) = \{\, \alpha \in \mathcal{K} : x \in \alpha \text{ and } g(x) = \max_{y \in \alpha} g(y) \,\}$$

give a disjoint partition of $\mathcal{K}$:

$$\mathcal{K} = \bigsqcup_{x \in D} L(x).$$

The following algorithm works on each lower star $L(x)$ in parallel. As a preparation, for each k-cell $\alpha = \{x, y_1, \ldots, y_{2^k-1}\} \in L(x)$, define

$$G(\alpha) = \big(g(x), g(y_{i_1}), \ldots, g(y_{i_{2^k-1}})\big) \in \mathbb{R}^{2^k},$$

where $g(x) > g(y_{i_1}) > \cdots > g(y_{i_{2^k-1}})$, and list these vectors lexicographically.

1. If $L(x) = \{x\}$, then the 0-cell x is marked as critical and end. Otherwise, the 0-cell x is paired with the 1-cell $\delta \in L(x)$ that is minimal with respect to G ordering.
2. Add all other 1-cells in $L(x)$ to the priority queue `PQzero`, whose elements are ordered by G.
3. Add all cells $\alpha \in L(x)$ such that α is a face of δ and the number of its unpaired faces is exactly one to the priority queue `PQone`, whose elements are ordered by G.
4. Iterate the following procedure while the queue `PQone` is not empty:

   - Pop the top cell α in `PQone`.

- If there are no unpaired faces of α, then add α to the queue `PQzero`. Otherwise, the k-cell α is paired with a $(k+1)$-cell β that is a single available unpaired face for the cell α. Remove the cell β from the queue `PQzero` and add all cells $\gamma \in L(x)$ such that γ is the face of both α and β and the number of its unpaired faces is exactly one to the queue `PQone`.

5. If the queue `PQzero` is empty, then end. Otherwise, pop the top cell γ in `PQzero` and mark γ as critical, add all cells $\alpha \in L(x)$ such that α is a face of γ and the number of its unpaired faces is exactly one to the queue `PQone`. Go to the step 4.

After executing the algorithm above for all the lower stars $L(x)$, all the cells in $\mathcal{K}$ are either marked as critical or paired with another cell. We can then define a discrete Morse function corresponding to g as follows. For each $x \in D$, if $L(x)$ has more than one cell, let δ be the 1-cell that is minimal with respect to G ordering. If $L(x)$ has $k > 2$ cells, let $\alpha_1, \alpha_2, \dots, \alpha_{k-2}$ be the remaining cells in $L(x)$, which are ordered by when they are marked or paired, where if the cells α and β are paired and β is a face of α then β will immediately precede α. Let $\epsilon = \min_{x \neq y}\{|g(x) - g(y)|\}$, then a discrete Morse function $m \colon \mathcal{K} \to \mathbb{R}$ is defined by

$$\begin{cases} m(\delta) = g(x) - \dfrac{\epsilon}{10}, \\ m(x) = g(x), \\ m(\alpha_i) = g(x) + \dfrac{i\epsilon}{10}, \quad i = 1, 2, \dots, k-2. \end{cases} \tag{2.1}$$

For this discrete Morse function, the pairings of cells yield its gradient vector field and the cells marked as critical become its critical cells.

Figure 2.6 illustrates examples of constructed discrete Morse functions and critical cells.

2.3.2 *Application to Analysis of the Burgers Cellular Automaton*

The Burgers cellular automaton [20] is a simple traffic flow model. Its time evolution equation is given by

$$U_n^{(t+1)} = U_n^{(t)} + \min\left(C - U_n^{(t)}, U_{n-1}^{(t)}\right) - \min\left(C - U_{n+1}^{(t)}, U_n^{(t)}\right),$$

where $n \in \mathbb{Z}$ denotes the site number, $t \in \mathbb{Z}$ is the discrete-time variable, and a positive constant $C \in \mathbb{Z}$ denotes the capacity of each site, i.e. $U_n^{(t)} \in \{0, 1, 2, \dots, C\}$.

First, let us consider the case of $C = 1$:

$$U_n^{(t+1)} = U_n^{(t)} + \min\left(1 - U_n^{(t)}, U_{n-1}^{(t)}\right) - \min\left(1 - U_{n+1}^{(t)}, U_n^{(t)}\right),$$

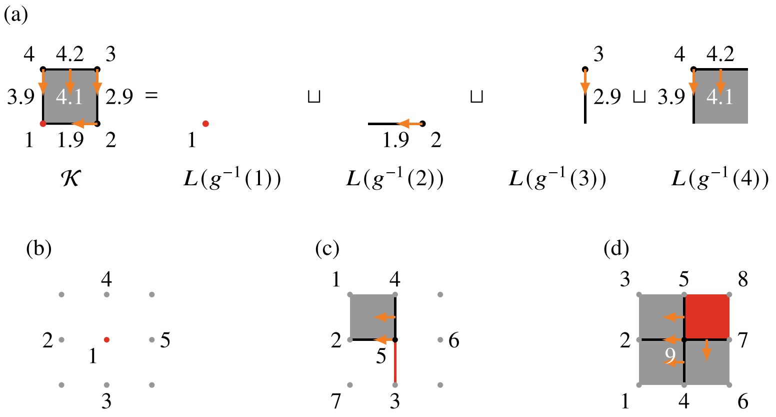


Fig. 2.6 Examples of constructed discrete Morse functions and critical cells. In each subfigure, numbers indicate the values of the original given functions or the constructed discrete Morse functions, orange arrows indicate the pairings of the cells in the algorithm or the gradient vector field, and red cells are critical, i.e. they have no paired cell. **a** The algorithm works on each lower star $L(x)$ in parallel. After pairing and marking, the values of a discrete Morse function are defined as equation (2.1). **b** An example of detecting a critical 0-cell, which is an analogue of a local minimum point in the continuous case. **c** An example of detecting a critical 1-cell, which is an analogue of a saddle point in the continuous case. **d** An example of detecting a critical 2-cell, which is an analogue of a local maximum point in the continuous case

which is also known as the elementary cellular automaton of rule 184 [27]. The value $U_n^{(t+1)}$ is determined by the values of 3-neighborhood $U_{n-1}^{(t)}$, $U_n^{(t)}$, and $U_{n+1}^{(t)}$:

$$\frac{U_{n-1}^{(t)}U_n^{(t)}U_{n+1}^{(t)}}{U_n^{(t+1)}} = \frac{111}{1}, \frac{110}{0}, \frac{101}{1}, \frac{100}{1}, \frac{011}{1}, \frac{010}{0}, \frac{001}{0}, \frac{000}{0}. \tag{2.2}$$

Note that $10111000_2 = 184_{10}$. We impose the periodic boundary condition $U_{n+K}^{(t)} = U_n^{(t)}$ for all n and a positive constant $K \in \mathbb{Z}$. Let us define the particle density ρ by

$$\rho = \frac{1}{CK}\sum_{n=0}^{K-1} U_n^{(t)}.$$

Note that ρ is a conserved quantity of the Burgers cellular automaton. Figure 2.7a and b show examples of the time evolution of the Burgers cellular automaton with $\rho > 0.5$ and $\rho < 0.5$, respectively.

.

For the data $\{U_n^{(t)}\}_{t=0}^{T-1}$, we consider the function g defined by

$$g(n,t) = U_n^{(t)} + \frac{(K-n)+Kt}{20K(T-t)}, \tag{2.3}$$

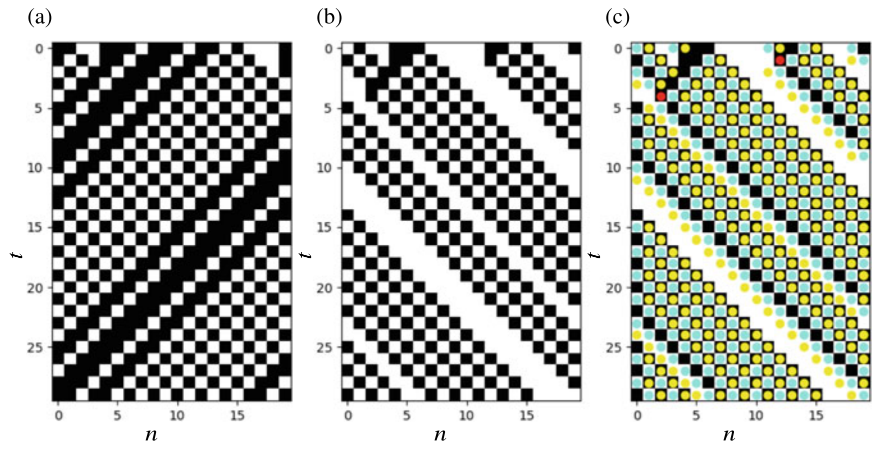


Fig. 2.7 Examples of the time evolution of the Burgers cellular automaton in the case of $C = 1$ and $K = 20$. White and black cells indicate $U_n^{(t)} = 0$ and 1, respectively. **a** In the case of $\rho > 0.5$, the steady state is a congested flow state. **b** In the case of $\rho < 0.5$, the steady state is a free flow state. **c** An example of applications of the discrete Morse theory to the states **b**. Blue, yellow, and red cells indicate that the lower star of the cell contains a critical 0-cell, a critical 1-cell, and a critical 2-cell, respectively

where the second term of the right-hand side is perturbation. Figure 2.7c shows an example of the result of the algorithm in the previous subsection for this function g. We can observe that critical 2-cells are detected at the points where a traffic jam disappears. This can be explained as follows. It is readily shown that the algorithm detects a critical 2-cell in a lower star $L(n, t)$ iff the value $g(n, t)$ is maximum in the Moore neighborhood of the point (n, t). In addition, under the definition (2.3),

- $g(\nu, \tau) < g(n, t)$ for $(\nu, \tau) = (n, t-1), (n+1, t-1), (n+1, t), (n+1, t+1)$ iff $U_\nu^{(\tau)} \leq U_n^{(t)}$.
- $g(\nu, \tau) < g(n, t)$ for $(\nu, \tau) = (n-1, t-1), (n-1, t), (n-1, t+1), (n, t+1)$ iff $U_\nu^{(\tau)} < U_n^{(t)}$.

Since $U_n^{(t)} \in \{0, 1\}$, the condition above implies that the lower star $L(n, t)$ contains a critical 2-cell iff

$$\begin{pmatrix} U_{n-1}^{(t-1)} & U_n^{(t-1)} & U_{n+1}^{(t-1)} \\ U_{n-1}^{(t)} & U_n^{(t)} & U_{n+1}^{(t)} \\ U_{n-1}^{(t+1)} & U_n^{(t+1)} & U_{n+1}^{(t+1)} \end{pmatrix} = \begin{pmatrix} 0 & * & * \\ 0 & 1 & * \\ 0 & 0 & * \end{pmatrix},$$

where $*$ allows either 0 or 1. However, from the time evolution rule of the Burgers cellular automaton (2.2), the $*$s above are uniquely determined as

$$\begin{pmatrix} U_{n-1}^{(t-1)} & U_n^{(t-1)} & U_{n+1}^{(t-1)} \\ U_{n-1}^{(t)} & U_n^{(t)} & U_{n+1}^{(t)} \\ U_{n-1}^{(t+1)} & U_n^{(t+1)} & U_{n+1}^{(t+1)} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

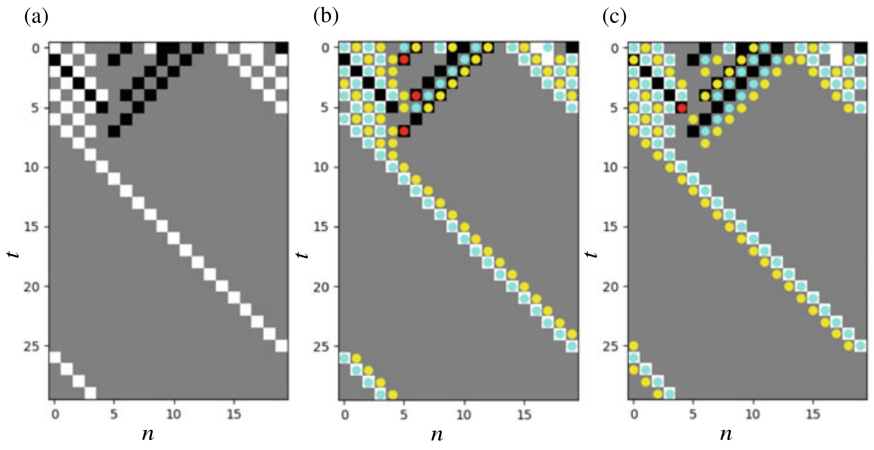


Fig. 2.8 **a** An example of the time evolution of the Burgers cellular automaton in the case of $C = 2$ and $K = 20$. White, gray, and black cells indicate $U_n^{(t)} = 0$, 1, and 2, respectively. **b** An example of applications of the discrete Morse theory to the states **a** using the function defined by (2.3). Blue, yellow, and red cells indicate that the lower star of the cell contains a critical 0-cell, a critical 1-cell, and a critical 2-cell, respectively. **c** An example using the function defined by (2.4) instead of (2.3). The meaning of color cells are same as in **b**

Therefore, a traffic jam disappears at the point (n, t) iff the lower star $L(n, t)$ contains a critical 2-cell.

For the case of $C = 2$, we can also show almost the same result in the same manner. See Fig. 2.8b. As another application, if we consider

$$g(n, t) = U_n^{(t)} + \frac{n + Kt}{20KT} \tag{2.4}$$

instead of (2.3), we can show in a similar manner that critical 2-cells are detected at the points where a free flow disappears. See Fig. 2.8c.

2.3.3 Application to Analysis of Pedestrian Flow

As a more practical application, we tried to apply the algorithm to the states of a pedestrian simulator developed by one of the authors, Zanlungo (see also Sect. 2.5). In the simulator, pedestrians are walking in a 2D map that has a crisscross corridor. Therefore the simulator generates $(2 + 1)$-dimensional (3D) data. From the data, we define a simple density function, pick its values on a lattice, construct the input function g with perturbation, and apply the algorithm for 3D lattice. Figure 2.9 shows an example of the result. At present, since this method detects a large number

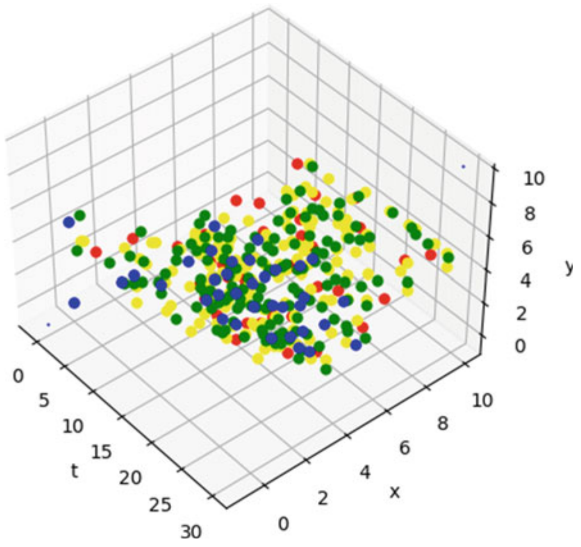


Fig. 2.9 An example of applications of the discrete Morse theory to the states of the pedestrian simulator by Zanlungo. Yellow, blue, green, and red cells indicate that the lower star of the cell contains a critical 0-cell, a critical 1-cell, a critical 2-cell, and a critical 3-cell, respectively

of critical cells, we are not able to interpret the result. It is left for future research to apply the discrete Morse theory not only to artificial simple data but also to practical data.

2.4 A Traffic Flow Model Using Quantum Walks

Quantum walk (QW), a quantum version of classical random walk, is widely applied to fast search algorithms by using quantum computers and modeling quantum systems [24]. Taking advantage of the flexibility of QWs, many extended QW models such as lattice models and many-particle models have been proposed [2, 24].

As discussed in the previous chapters, classical (non-quantum) particle models including cell automata have the potential to represent phenomena of traffic flow, and their properties have been studied for a long time. In real-world traffic flows, featured phenomena such as free running and congested phases and traffic vibration caused by repeated low-speed and high-speed states are known [16].

Although quantum computing has been widely applied to computational acceleration and modeling, quantum models have limited modeling of traffic flows. To investigate the applicability of quantum models to traffic flows, this section discusses how to construct a quantum model of traffic flows using QW, named one-way multi-particle quantum walks (OMQW). Also, by simulation, this section presents the behaviors of OMQW and displays the basic diagram of OMQW by using the "quantity of flow".

2.4.1 A Definition of One-Way Multi-particle Quantum Walks

In this part, we introduce n-particle discrete-time OMQW to discuss the quantum traffic flow model. Intuitively speaking, OMQW has two differences from standard discrete-time quantum walks. One is that it is composed of multiple particles (n particles), and the other is to introduce unidirectionality to express the flow. Several quantum models composed of many particles have already been reported, and OMQW now adopts the same approach as Costa et al to handle multiple particles [8]. They defined the interaction between particles by collision to consider the quantum version of gas model, i.e. collisions between particles occur if multiple particles arrive at the same coordinates as a result of movement. Their collision formulation is suitable for quantum models because it is unitary. Next, to realize unidirectionality, OMQW is a model which expands a quantum walk with two internal degrees of freedom, “move forward” and “stay”. These two internal degrees of freedom are inspired by TASEP, where these two actions are selected stochastically. Contrastingly, OMQW treats them as states (of the coin) by means of quantum.

Before defining OMQW, we describe the definition of symbols related to OMQW. Let $\mathcal{H}_p$ be a Hilbert space and $\mathcal{H}_c$ be a finite dimensional Hilbert space. Let us denote an orthonormal basis in $\mathcal{H}_p$ by $\{|x\rangle\}_{x\in\mathbb{Z}}$, and an orthonormal basis in $\mathcal{H}_c$ by $\{|c\rangle\}_{c\in\{0,1\}}$. Then, an orthonormal basis of $\mathcal{H}_p \otimes \mathcal{H}_c$ can be written as $\{|x, c\rangle := |x\rangle \otimes |c\rangle\}_{x\in\mathbb{Z},c\in\{0,1\}}$. By expanding these notation to n-particle, orthonormal set in $\hat{\mathcal{H}} := \bigotimes_{i=1}^{n}(\mathcal{H}_p \otimes \mathcal{H}_c)$ can be written as $\{|\boldsymbol{x}, \boldsymbol{c}\rangle\}_{\boldsymbol{x}=(x_1,\dots,x_n)\in\mathbb{Z}^n,\boldsymbol{c}=(c_1,\dots,c_n)\in\{0,1\}^n}$.

OMQW is represented by operators on $\hat{\mathcal{H}}$, because the state transition of OMQW is identified by behaviors of n particles and n coins. More concretely, one step of OMQW consists of three operators: an n-particle collision operator $\hat{T}$, a shift operator $\hat{S}$, and coin operators $\hat{C}_{\boldsymbol{d}}$. The exact definitions of these three operators are a bit complicated, so the overall OMQW definition is described first, and then define these operators.

Definition 2.1 (*One-Way Multi-particle Quantum Walks*) If any states $|\psi_t\rangle \in \hat{\mathcal{H}}$ at time $t \in \mathbb{Z}_{\geq 1}$ satisfy the following conditions, we say that a dynamics of $|\psi_t\rangle$ is one-way multi-particle quantum walk (OMQW).

1. (Initial condition) The initial state $|\psi_0\rangle$ is given.
2. (Deterministic initial position) Let $\boldsymbol{s} \in \mathbb{Z}^n$ be initial position. For any $\boldsymbol{x} \in \mathbb{Z}^n$, $\boldsymbol{c} \in \{0, 1\}^n$, if $\boldsymbol{x} \neq \boldsymbol{s}$, then $\|\langle \boldsymbol{x}, \boldsymbol{c}|\psi_0\rangle\|^2 = 0$ holds.
3. (State transition) The state transition is given by

$$|\psi_t\rangle = \sum_{\boldsymbol{d}\in\{0,1\}^n} \hat{T}\hat{S}\hat{C}_{\boldsymbol{d}}|\psi_{t-1}\rangle \tag{2.5}$$

In the rest of this section, we define these three operators consisting of OMQW and describe their properties.

2.4.1.1 Coin Operator

First of all, let us introduce a coin matrix for one particle $D \in U(2)$, where U means unitary group. We denote the i-th row j-th column element in D by $\alpha_{i,j} \in \mathbb{C}$. Note that the index of this matrix starts at zero, i.e. $i, j \in 0, 1$.

Let $d \in \{0, 1\}$ be an index representing the next state of a coin. A coin operator for a single particle $C_d : \mathcal{H}_p \otimes \mathcal{H}_c \to \mathcal{H}_p \otimes \mathcal{H}_c$ is defined as: $C_d := \sum_{\substack{x\in\mathbb{Z}\\ c\in\{0,1\}}} \alpha_{d,c} |x, d\rangle\langle x, c|$. Intuitively, this operation represents the change of the coin state from the current state c to the next state d.

The next definition describes the coin operator for multiple particles by extending the above coin operator for a single particle.

Definition 2.2 (*Coin operator*) For any $\boldsymbol{d} \in \{0, 1\}^n$, n-particle coin operator $\hat{C}_{\boldsymbol{d}} : \hat{\mathcal{H}} \to \hat{\mathcal{H}}$ is defined as:

$$\hat{C}_{\boldsymbol{d}} := \sum_{\substack{x_1,\dots,x_n\in\mathbb{Z}\\ c_1,\dots,c_n\in\{0,1\}}} \bigotimes_{i=1}^{n} (C_{d_i} |x_i, c_i\rangle\langle x_i, c_i|) \tag{2.6}$$

The following properties of the one-/n-particle coin operator hold.

- For any $\boldsymbol{d} \in \{0, 1\}^n$, $x \in \mathbb{Z}$, and $c \in \{0, 1\}$, the following equation holds:

$$C_d|x, c\rangle = \alpha_{d,c}|x, d\rangle \tag{2.7}$$

- For any $\boldsymbol{d} \in \{0, 1\}^n$, $\boldsymbol{x} \in \mathbb{Z}^n$, and $\boldsymbol{c} \in \{0, 1\}^n$, the following equation holds:

$$\hat{C}_{\boldsymbol{d}}|\boldsymbol{x}, \boldsymbol{c}\rangle = \bigotimes_{i=1}^{n} (C_{d_i}|x_i, c_i\rangle) = (\prod_{i=1}^{n} \alpha_{d_i,c_i})|\boldsymbol{x}, \boldsymbol{c}\rangle \tag{2.8}$$

2.4.1.2 Shift Operator

The shift operator represents the one-time step movement of particles depending on the state of the coin.

Definition 2.3 (*Shift operator*) A one-particle shift operator $S : \mathcal{H}_p \otimes \mathcal{H}_c \to \mathcal{H}_p \otimes \mathcal{H}_c$ is defined as: $S := \sum_{\substack{x\in\mathbb{Z}\\ c\in\{0,1\}}} |x + c, c\rangle\langle x, c|$

An n-particle shift operator $\hat{S} : \hat{\mathcal{H}} \to \hat{\mathcal{H}}$ is defined as:

$$\hat{S} := \sum_{\substack{x_1,\dots,x_n\in\mathbb{Z}\\ c_1,\dots,c_n\in\{0,1\}}} \bigotimes_{i=1}^{n} (S|x_i, c_i\rangle\langle x_i, c_i|) \tag{2.9}$$

Then, the one-particle/n-particle shift operator satisfies the following two statements.

- For any $x \in \mathbb{Z}$ and $c \in \{0, 1\}$, the following equation holds: $S|x, c\rangle = |x + c, c\rangle$
- For any $\boldsymbol{x} = (x_1, \dots, x_n) \in \mathbb{Z}^n$ and $\boldsymbol{c} = (c_1, \dots, c_n) \in \{0, 1\}^n$, the following equation holds:

$$\hat{S}|\boldsymbol{x}, \boldsymbol{c}\rangle = \bigotimes_{i=1}^{n}(S|x_i, c_i\rangle) = |\boldsymbol{x} + \boldsymbol{c}, \boldsymbol{c}\rangle \tag{2.10}$$

2.4.1.3 Collision Operator

Before defining the collision operator, we introduce a collision function that represents whether a particle collides with another particle. Let $i \in \{1, \dots, n\}$ be an index of particle. Collision function $r_i : \mathbb{Z}^n \times \{0, 1\} \to \{0, 1\}$ is defined as

$$r_i(\boldsymbol{x}, c) := \begin{cases} 1 - c & (\sum_{j=1}^{n} \delta_{x_i, x_j} \quad is \quad even) \\ c & (\sum_{j=1}^{n} \delta_{x_i, x_j} \quad is \quad odd) \end{cases} \tag{2.11}$$

where $\delta_{\cdot,\cdot}$ is Kronecker delta. By extending this function to n-particles, a collision function for n-particles $\hat{\boldsymbol{r}} : \mathbb{Z}^n \times \{0, 1\}^n \to \{0, 1\}^n$ is defined as

$$\hat{\boldsymbol{r}}(\boldsymbol{x}, \boldsymbol{c})_i := r_i(\boldsymbol{x}, c_i) \tag{2.12}$$

where $\hat{\boldsymbol{r}}(\boldsymbol{x}, \boldsymbol{c})_i$ is i-th element of $\hat{\boldsymbol{r}}(\boldsymbol{x}, \boldsymbol{c})$.

Definition 2.4 (*Collision operator*) n-particle collision operator $\hat{T} : \hat{\mathcal{H}} \to \hat{\mathcal{H}}$ is defined as

$$\hat{T} := \sum_{\substack{x_1, \dots, x_n \in \mathbb{Z} \\ c_1, \dots, c_n \in \{0,1\}}} \bigotimes_{i=1}^{n}(|x_i, r_i(\boldsymbol{x}, c_i)\rangle\langle x_i, c_i|) \tag{2.13}$$

The collision operator describes the change of internal states ("move forward" and "stay") when particles collide. Typically, when two particles are at the same coordinates, the state of the particles is flipped.

The collision operator and function have the following properties:

- For $i \in \{1, \dots, n\}$ and $\boldsymbol{x} \in \mathbb{Z}^n$, $r_i(\boldsymbol{x}, \cdot)$ is bijective.
- For $\boldsymbol{x} \in \mathbb{Z}^n$, $\hat{\boldsymbol{r}}(\boldsymbol{x}, \cdot)$ is bijective.
- For $\boldsymbol{x} \in \mathbb{Z}^n$, $\boldsymbol{c} \in \{0, 1\}^n$, applying a collision operator is represented as form:

$$\hat{T}|\boldsymbol{x}, \boldsymbol{c}\rangle = |\boldsymbol{x}, \hat{\boldsymbol{r}}(\boldsymbol{x}, \boldsymbol{c})\rangle \tag{2.14}$$

- For $\boldsymbol{x} \in \mathbb{Z}^n$, and $\boldsymbol{c} \in \{0, 1\}^n$, the following equation holds:

$$\hat{\boldsymbol{r}}(\boldsymbol{x}, \hat{\boldsymbol{r}}(\boldsymbol{x}, \boldsymbol{c})) = \boldsymbol{c} \tag{2.15}$$

2.4.2 Probability Distribution of OMQW

When the positions of particles in OMQW are observed after t-step time evolution, the observation behaves stochastically due to their quantum nature. This quantum property of OMQW is derived from the standard single-particle one-dimensional QW. The standard QW does not have a property called locality, where probabilities concentrate at constant points as time goes to infinity, unlike classical random walks. In this section, to confirm the behavior related to the non-locality of OMQW experimentally, we describe the formulation of the probability distribution of positions of particles and a computational method of such a probability.

From the quantum nature, positions $\boldsymbol{x}$ and internal states $\boldsymbol{c}$ of particles are stochastically observed, and their joint probability at time t is described as follows: $\|\langle \boldsymbol{x}, \boldsymbol{c}|\psi_t\rangle\|^2$. Note that, by the definition related to deterministic initial positions of OMQW, observation at $t = 0$ should be observed at position s.

2.4.2.1 Computation of Probability

In this section, we derive a recursive formula to compute the probabilities of states of OMQW. To express the state on the computer specifically, coefficients of the state $a_{t,\boldsymbol{x},\boldsymbol{c}} \in \mathbb{C}$ are introduced as $a_{t,\boldsymbol{x},\boldsymbol{c}} := \langle \boldsymbol{x}, \boldsymbol{c}|\psi_t\rangle$. Since $\{|\boldsymbol{x}, \boldsymbol{c}\rangle\}_{\boldsymbol{x}\in\mathbb{Z}^n, \boldsymbol{c}\in\{0,1\}^n}$ is an orthonormal set on $\hat{\mathcal{H}}$, a state $|\psi_t\rangle$ can be represented as follows:

$$|\psi_t\rangle = \sum_{\substack{\boldsymbol{x}\in\mathbb{Z}^n \\ \boldsymbol{c}\in\{0,1\}^n}} a_{t,\boldsymbol{x},\boldsymbol{c}} |\boldsymbol{x}, \boldsymbol{c}\rangle \tag{2.16}$$

By combining these coefficients with the definitions in the previous section, the following recursive formula with respect to $a_{t,\boldsymbol{x},\boldsymbol{c}}$ can be deduced.

Theorem 2.1 (Recursive expression for $a_{t,\boldsymbol{x},\boldsymbol{c}}$) *For any $t \in \mathbb{Z}_{\geq 1}, \boldsymbol{x} \in \mathbb{Z}^n$, and $\boldsymbol{c} \in \{0, 1\}^n$, the following equation holds:*

$$a_{t,\boldsymbol{x},\boldsymbol{c}} = \sum_{\tilde{\boldsymbol{c}}\in\{0,1\}^n} a_{t-1,\boldsymbol{x}-\hat{\boldsymbol{r}}(\boldsymbol{x},\boldsymbol{c}),\tilde{\boldsymbol{c}}} (\prod_{i=1}^{n} \alpha_{r_i(\boldsymbol{x},c_i),\tilde{c}_i}) \tag{2.17}$$

Note that, the reachable area from the initial position in t steps is limited by t. Let the reachable area be $R(t) \subset Z^n$. If $x \in Z^n \setminus R(t)$, then $a_{t,\boldsymbol{x},\boldsymbol{c}} = 0$.

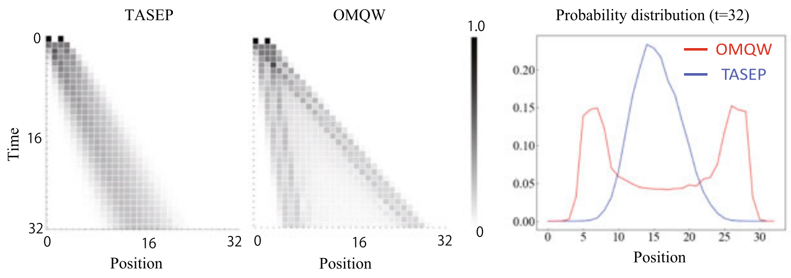


Fig. 2.10 This figure shows the time evolution of TASEP (left) and OMQW (center) where x-axis is position and y-axis is time step. Color intensity indicates the expected number of particles present at the position. The right figure shows the distributions of particles after 32 steps, these are corresponding with the bottom line of left and center figures

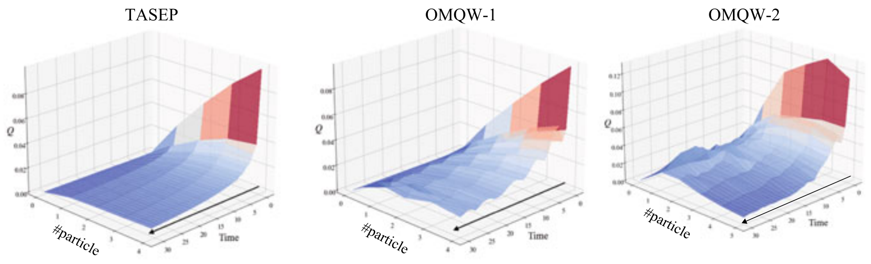


Fig. 2.11 Fundamental diagrams: quantity of flow related to the number of particles and time evolution for TASEP and OMQW (OMQW-1). OMQW-2 is experiments for an under periodic boundary conditions

This theorem can be proven by mathematical induction. By using this theorem, we can computationally simulate the state after any steps by performing recursively calculating with the above formula.

2.4.3 Simulation

This section carried out computer simulations on OMQW and TASEP using the definition until previous section (Fig. 2.10). These figures show the distributions of OMQW and TASEP, which are computed by simulation at 1000 times for each time step. The TASEP result shows a unimodal distribution related to the particle positions, whose center moves to the right. Contrastingly, the distribution of OMQW dividing into two modalities: slow and fast modal. A normal quantum walk is known to have two distribution peaks, which are the same as OMQW. A difference point is that OMQW constitutes a flow due to the introduction of unidirectionality.

Next, we introduce "quantity of flow" to discuss its properties as traffic flow. Let be position x and be time t. Quantity of flow Q_t is defined as

$$Q_t = 1/N \sum_{0}^{N-1} \max(E_{x,t} - E_{x,t-1}, 0)$$

where $E_{x,t}$ is expectation of the number of particles at time step t in OMQW, the initial position is set at random.

Figure 2.11 shows the relationship between this quantity, x, and t. Comparing the results of TASEP and those of OMQW, it can be seen that the quantity of flow of OMQW is oscillates related to time t. These properties, unlike classical models, might be used for modeling phenomena in traffic flow.

2.4.3.1 Summary

In this section, we discussed applicability of quantum walk to a traffic model. We introduced an OMQW model, which is an extension of QW to a traffic flow model. In the first experiment, we visualized the flow rate and clarified the basic properties of OMQW. In the future, we plan to take advantage of this property and extend it to a more realistic and applied model.

2.5 Pedestrian Flow and Future Perspectives

Understanding the dynamics of pedestrian crowds is of fundamental importance in a plethora of practical applications related to modern life, in particular in large urban areas, such as the planning of large-scale events, buildings, transportation hubs [14], or whole urban areas [3], both in normal and emergency situations (e.g. during natural disasters, fires, or even during pandemics [7]). While most of these applications focus on the macroscopic (large-scale) aspects of crowd dynamics, also the understanding of how pedestrians behave at the microscopic (individual) level has considerable practical relevance, in particular to the field of automatic navigation of cars, robots, and other vehicles in urban areas where pedestrians may be present or even dominant [23, 36], and to safety and surveillance [25]. Furthermore, improved understanding of microscopic behavior may reflect on the ability of reproducing macroscopic dynamics, as the latter may be strongly influenced by individual dynamics such as those determined by belonging to a social group [28, 30, 34, 35].

Along with the aforementioned practical relevance, crowd dynamics is also of extreme theoretical importance, as it may be described as a many-particle system (and, in the continuum limit, as a fluid) whose fundamental components are human beings, and thus represents a natural arena to try to apply the mathematical methods of statistical physics to social systems. Namely, while the relevant observables of the problems are the same as in a classical physics system (density, fluid velocity, pressure at the macroscopic level, position velocity, acceleration, orientation at the microscopic level), the laws determining the dynamics are related to human behavior, with all the related challenges that this implies [13].

The understanding of crowd dynamics relies mainly on three related, but conceptually different, scientific challenges: (1) data collection, (2) modeling, and (3) data analysis. All three fields have seen considerable improvements in recent years. It is nowadays possible to detect and track pedestrian positions in real time, allowing to collect a large amount of information concerning pedestrian behavior in controlled [5] and “ecological” [26, 29] settings. These data have obviously inspired mathematical and computational models of pedestrian and crowd behavior, and allowed to calibrate such models [13].

In this project we are mainly interested in the third challenge, data analysis. Pedestrian flow presents obviously many common points with the vehicular flow, analyzed in Sect. 2.3, but it is inherently more complex than the former. By being less strictly regulated than vehicular traffic, and composed of basic units (pedestrians) whose motion is less dynamically constrained than that of cars, pedestrian flow is a completely 2-dimensional phenomenon, while vehicular flow presents strong 1-dimensional features, and thus its topological properties are more complex.

As a result, practical and theoretical applications of discrete Morse theory to pedestrian flows are expected to be developed only after the relevant vehicular ones, and for this reason in the present section, we necessarily report a research plan more than actual accomplishments. Nevertheless, we believe that the contribution of the proposed framework to the study of crowd dynamics and pedestrian behavior may be extremely relevant and promising.

While as stated above the pedestrian flow problem is inherently 2-dimensional, most literature works focus on simplified settings that basically reduce it to a 1-dimensional one, such as corridors or bottlenecks [1, 9, 10, 19, 22, 37], and as a result also the related analysis tools are based on observables that do not rely on the 2-dimensional structure of space, such as pedestrian density and average speed.

On the other hand, a recent work [31] has introduced a novel metric, congestion number CN, to assess the state of a pedestrian crowd. This metric, being based on the magnitude of the gradient of the only non-zero component (i.e. normal to the floor) of the rotor of the pedestrian velocity

$$\|\nabla(\nabla \times \mathbf{v})_z\| \tag{2.18}$$

(divided by a theoretical reference value to obtain a pure number), is more apt to study actual 2-dimensional phenomena, and has proved to provide additional information with respect to the traditional 1-dimensional tools.

The aforementioned theoretical tool has been applied also to a geometrical setting that, although being still extremely simple if compared to the complexity of the real world, is fully two-dimensional, namely the cross-flow problem, i.e. the dynamics of two different pedestrian streams crossing at a non-trivial angle $\theta \neq n\pi$ ($\theta = n\pi$ corresponding to the well-studied corridor case).

This problem, and in particular the related formation of self-organized patterns (stripes) in the crossing area, has been the object of a few recent contributions, focusing on how the dynamics are affected by the angle θ [18] and by pedestrian density [32], on the ability of different collision avoidance models to reproduce the observed dynamics [33], and on a mathematical model of the self-organization phenomenon [4].

We believe the cross-flow scenario to be a benchmark to test the potential of the discrete Morse theory in studying flow properties that go beyond the vehicular applications analyzed in Sect. 2.3. We first of all aim to develop a way to use the discrete Morse theory to identify criticalities in the cross-flow scenario that may go beyond an analysis based only on density and speed, in a way similar to the results achieved by [31].

After this first goal is achieved, the method may be applied to still poorly understood phenomena. The ultimate challenge is to apply the method to the study of real-world data, possibly to understand which geometrical settings and crowd conditions may lead to dangerous circumstances and possibly to casualties.

While this is an extremely difficult task, an intermediate goal may be to study more complex versions of the basic cross-flow scenario, i.e. by increasing the number of flows or introducing obstacles [6].

While, as stated above, this project is aimed at tackling the challenge of "data analysis", such a task cannot be independent of data collection and modeling. We will use and further develop the computational models of [33] to obtain artificial data that may be used to test the proposed discrete Morse theory-based analysis on different crowd settings.

Obviously, while artificial data have the merit of being easily and safely produced and analyzed, data concerning actual pedestrian behavior are indispensable to develop an analysis tool that aims to describe the real world. While in this project we will mainly rely on the increasing number of freely available data concerning controlled experiments and real-world behavior, we do not discard the possibility of performing some simple experiments with human participants explicitly aimed at the development of the proposed analysis methods.

References

1. J. Adrian, A. Seyfried, A. Sieben, Crowds in front of bottlenecks at entrances from the perspective of physics and social psychology. J. R. Soc. Interface **17**(165), 20190871 (2020)
2. A. Ahlbrecht, A. Alberti, D. Meschede, V.B. Scholz, A.H. Werner, R.F. Werner, Molecular binding in interacting quantum walks. New J. Phys. **14**(7), 073050 (2012)
3. S.M. Arisona, G. Aschwanden, J. Halatsch, P. Wonka, *Digital Urban Modeling and Simulation* (Springer, 2012)
4. K.A. Bacik, B.S. Bacik, T. Rogers, Lane nucleation in complex active flows. Science **379**(6635), 923–928 (2023)
5. M. Boltes, A. Seyfried, Collecting pedestrian trajectories. Neurocomputing **100**, 127–133 (2013)

6. S. Cao, A. Seyfried, J. Zhang, S. Holl, W. Song, Fundamental diagrams for multidirectional pedestrian flows. J. Stat. Mech.: Theory Exp. **2017**(3), 033404 (2017)
7. S. Comai, S. Costa, S.M. Mastrolembo Ventura, G. Vassena, L. Tagliabue, D. Simeone, E. Bertuzzi, G. Scurati, F. Ferrise, A. Ciribini, Indoor mobile mapping system and crowd simulation to support school reopening because of COVID-19: a case study, in *Proceedings of the 13th GeoInformation for Disaster Management Conference* (2020), pp. 29–36
8. P.C. Costa, F. De Melo, R. Portugal, Multiparticle quantum walk with a gas like interaction. Phys. Rev. A **100**(4), 042320 (2019)
9. C. Feliciani, K. Nishinari, Empirical analysis of the lane formation process in bidirectional pedestrian flow. Phys. Rev. E **94**(3), 032304 (2016)
10. C. Feliciani, I. Zuriguel, A. Garcimartín, D. Maza, K. Nishinari, Systematic experimental investigation of the obstacle effect during non-competitive and extremely competitive evacuations. Sci. Rep. **10**(1), 1–20 (2020)
11. R. Forman, Morse theory for cell complexes. Adv. Math. **134**, 90–145 (1998)
12. R. Forman, A user's guide to discrete morse theory. Sémin. Lothar. Comb. **48**, B48c (2002)
13. D. Helbing, P. Molnar, Social force model for pedestrian dynamics. Phys. Rev. E **51**(5), 4282 (1995)
14. G. Hoy, E. Morrow, A. Shalaby, Use of agent-based crowd simulation to investigate the performance of large-scale intermodal facilities: case study of union station in Toronto, Ontario. Canada. Transp. Res. Rec. **2540**(1), 20–29 (2016)
15. T. Kaczynski, K. Mischaikow, M. Mrozek, *Computational Homology* (Springer, 2004)
16. X. Li, F. Peng, Y. Ouyang, Measurement and estimation of traffic oscillation properties. Transp. Res. Part B: Methodol. **44**(1), 1–14 (2010)
17. J. Milnor, Morse theory. Ann. Math. **51** (1963)
18. P. Mullick, S. Fontaine, C. Appert-Rolland, A.-H. Olivier, W.H. Warren, J. Pettré, Analysis of emergent patterns in crossing flows of pedestrians reveals an invariant of "stripe" formation in human data. PLOS Comput. Biol. **18**(6), e1010210 (2022)
19. H. Murakami, C. Feliciani, Y. Nishiyama, K. Nishinari, Mutual anticipation can contribute to self-organization in human crowds. Sci. Adv. **7**(12), eabe7758 (2021)
20. K. Nishinari, D. Takahashi, Analytical properties of ultradiscrete Burgers equation and rule-184 cellular automaton. J. Phys. A: Math. Gen. **31**, 5439–5450 (1998)
21. V. Robins, P.J. Wood, A.P. Sheppard, Theory and algorithms for constructing discrete morse complexes from grayscale digital images. IEEE Trans. Pattern Anal. Mach. Intell. **33**(7), 1646–1658 (2011)
22. A. Seyfried, O. Passon, B. Steffen, M. Boltes, T. Rupprecht, W. Klingsch, New insights into pedestrian flow through bottlenecks. Transp. Sci. **43**(3), 395–406 (2009)
23. M. Shiomi, F. Zanlungo, K. Hayashi, T. Kanda, Towards a socially acceptable collision avoidance for a mobile robot navigating among pedestrians using a pedestrian model. Int. J. Robot. Res. **6**(3), 443–455 (2014)
24. S.E. Venegas-Andraca, Quantum walks: a comprehensive review. Quantum Inf. Process. **11**(5), 1015–1106 (2012)
25. X. Wang, M. Wang, W. Li, Scene-specific pedestrian detection for static video surveillance. IEEE Trans. Pattern Anal. Mach. Intell. **36**(2), 361–374 (2013)
26. J. Willems, A. Corbetta, V. Menkovski, F. Toschi, Pedestrian orientation dynamics from high-fidelity measurements. Sci. Rep. **10**(1), 1–10 (2020)
27. S. Wolfram, Computation theory of cellular automata. Commun. Math. Phys. **96**, 15–57 (1984)
28. Z. Yücel, F. Zanlungo, M. Shiomi, Modeling the impact of interaction on pedestrian group motion. Adv. Robot **32**(3), 137–147 (2018)
29. F. Zanlungo, D. Brščić, T. Kanda, Spatial-size scaling of pedestrian groups under growing density conditions. Phys. Rev. E **91**(6), 062810 (2015)
30. F. Zanlungo, L. Crociani, Z. Yücel, T. Kanda, The effect of social groups on the dynamics of bi-directional pedestrian flow: a numerical study. Traffic Granul. Flow **2019**, 307–313 (2020)
31. F. Zanlungo, C. Feliciani, Z. Yücel, X. Jia, K. Nishinari, T. Kanda, A pure number to assess "congestion" in pedestrian crowds. Transp. Res. Part C: Emerg. Technol. **148**, 104041 (2023)

32. F. Zanlungo, C. Feliciani, Z. Yücel, K. Nishinari, T. Kanda, Macroscopic and microscopic dynamics of a pedestrian cross-flow: Part I, experimental analysis. Saf. Sci. **158**, 105953 (2023)
33. F. Zanlungo, C. Feliciani, Z. Yücel, K. Nishinari, T. Kanda, Macroscopic and microscopic dynamics of a pedestrian cross-flow: Part II, modelling. Saf. Sci. **158**, 105969 (2023)
34. F. Zanlungo, T. Ikeda, T. Kanda, Potential for the dynamics of pedestrians in a socially interacting group. Phys. Rev. E **89**(1), 012811 (2014)
35. F. Zanlungo, Z. Yücel, D. Brščić, T. Kanda, N. Hagita, Intrinsic group behaviour: dependence of pedestrian dyad dynamics on principal social and personal features. PLOS ONE **12**(11), e0187253 (2017)
36. F. Zanlungo, Z. Yücel, F. Ferreri, J. Even, L.Y. Morales Saiki, T. Kanda, Social group motion in robots, in *Proceedings of the 9th International Conference on Social Robotics* (2017), pp. 474–484
37. J. Zhang, W. Klingsch, A. Schadschneider, A. Seyfried, Ordering in bidirectional pedestrian flows and its influence on the fundamental diagram. J. Stat. Mech.: Theory Exp. **2012**(02), P02002 (2012)

Chapter 3
Integrable Systems Related to Matrix LR Transformations

Masashi Iwasaki, Masato Shinjo, Yusaku Yamamoto, Akiko Fukuda, Sennosuke Watanabe, Masaki Sekiguchi, and Emiko Ishiwata

3.1 Introduction

The definition of integrable systems has been the subject of various discussions. However, in applied mathematics, integrable systems can be regarded as a general term for systems that precisely grasp some concept, as well as for systems whose solutions can be written explicitly. One important development in the study of such integrable systems is soliton theory [4, 5, 12], which made great strides in the latter

M. Iwasaki (✉)
Kyoto Prefectural University, 1-5 Nakaragi-cho, Shimogamo, Sakyo-ku, Kyoto 606-8522, Japan
e-mail: imasa@kpu.ac.jp

M. Shinjo
Osaka Seikei University, 1-3-7 Aikawa, Higashiyodogawa-ku, Osaka 533-0007, Japan
e-mail: shinjo@osaka-seikei.ac.jp

Y. Yamamoto
The University of Electro-Communications, 1-5-1 Chofugaoka, Chofu-shi, Tokyo 182-8585, Japan
e-mail: yusaku.yamamoto@uec.ac.jp

A. Fukuda
Shibaura Institute of Technology, 307 Fukasaku, Minuma-ku, Saitama-shi, Saitama 337-8570, Japan
e-mail: afukuda@shibaura-it.ac.jp

S. Watanabe
The University of Fukuchiyama, 3370 Azahori, Fukuchiyama-shi, Kyoto 620-0886, Japan
e-mail: sewatana@fukuchiyama.ac.jp

M. Sekiguchi
Tokyo Metropolitan Ogikubo High School, 5-7-20 Ogikubo, Suginami-ku, Tokyo 167-0051, Japan
e-mail: emafarms@gmail.com

E. Ishiwata
Tokyo University of Science, 1-3 Kagurazaka, Shinjuku-ku, Tokyo 162-8601, Japan
e-mail: ishiwata@rs.tus.ac.jp

K. Ikeda et al. (eds.), *Advanced Mathematical Science for Mobility Society*,
https://doi.org/10.1007/978-981-99-9772-5_3

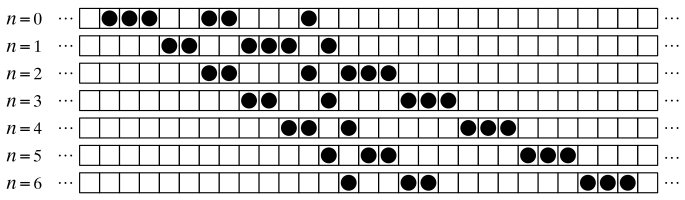


Fig. 3.1 An example of discrete-time evolution from $n = 0$ to $n = 6$ of the BBS

half of the twentieth century. During an interaction, soliton waves change their shape and velocity when the fast wave overtakes the slow wave. However, after an interaction, both waves return to their pre-interaction states. Soliton equations that represent these interactions can be discretized with respect to the time variable by skillful manners specific to integrable systems. The motions of soliton waves are typically drawn on a computer. The discrete soliton equations can be further simplified by quantization on the velocity of the soliton. This procedure, called the ultradiscretization, has resulted in the so-called box-and-ball systems (BBSs) [31], a type of cellular automaton. Cellular automata summarize various discrete phenomena by defining state-change rules for cells packed in a lattice. For example, they are useful for analyzing traffic flows to capture the motion of multiple cars. Therefore, we expect that BBSs will enrich the mathematical representation of mobility phenomena.

In the first proposed BBS [31], each ball, in order from left to right, moves to the nearest empty box to its right along an infinite number of boxes, arranged in a straight line without gaps. An example of the discrete-time evolutions of this BBS is shown in Fig. 3.1. Interestingly, the velocity of each soliton translates into the number of successive balls in each ball group, which can be determined by observation. We can describe the motion of m ball groups under a discrete-time evolution from n to $n+1$ using the ultradiscrete Toda (udToda) equation:

$$\begin{cases} Q_k^{(n+1)} = \min\left(\sum_{i=1}^{k} Q_i^{(n)} - \sum_{i=1}^{k-1} Q_i^{(n+1)}, E_k^{(n)}\right), \quad k = 1, 2, \ldots, m, \\ Q_k^{(n+1)} + E_k^{(n+1)} = Q_{k+1}^{(n)} + E_k^{(n)}, \quad k = 1, 2, \ldots, m-1, \\ E_0^{(n)} := +\infty, \quad E_m^{(n)} := +\infty, \end{cases} \tag{3.1}$$

where $Q_k^{(n)}$ and $E_k^{(n)}$ are the number of successive balls in the kth ball group from the left and the number of successive boxes in the box array between the kth and $(k+1)$th ball groups at discrete-time n, respectively. The udToda equation (3.1) is derived from the ultradiscretization of the discrete Toda (dToda) equation [13]:

$$\begin{cases} q_k^{(n+1)} + e_{k-1}^{(n+1)} = q_k^{(n)} + e_k^{(n)}, \quad k = 1, 2, \ldots, m, \\ q_k^{(n+1)} e_k^{(n+1)} = q_{k+1}^{(n)} e_k^{(n)}, \quad k = 1, 2, \ldots, m-1, \\ e_0^{(n)} := 0, \quad e_m^{(0)} := 0, \end{cases} \tag{3.2}$$

which is a famous discrete integrable system and has branches in fields other than mathematical physics. The soliton behavior in the BBS ascribes conserved quantities that remain unchanged under the discrete-time evolution in the udToda equation (3.1). Of course, these conserved quantities are derived from the dToda equation (3.2). We can capture conserved quantities of the dToda equation (3.2) in the following matrix representation that the dToda equation (3.2) satisfies, which is called the Lax representation:

$$L^{(n+1)}R^{(n+1)} = R^{(n)}L^{(n)}, \tag{3.3}$$

$$L^{(n)} := \begin{bmatrix} 1 & & & \\ e_1^{(n)} & 1 & & \\ & \ddots & \ddots & \\ & & e_{m-1}^{(n)} & 1 \end{bmatrix}, \quad R^{(n)} := \begin{bmatrix} q_1^{(n)} & 1 & & \\ & q_2^{(n)} & \ddots & \\ & & \ddots & 1 \\ & & & q_m^{(n)} \end{bmatrix}. \tag{3.4}$$

Using the equality of each entry in the Lax representation (3.3), we can easily obtain the dToda equation (3.2). Note that $R^{(n)}$ is nonsingular. Consider $A^{(n)} := L^{(n)}R^{(n)}$ as a tridiagonal matrix. Then we obtain $A^{(n+1)} = R^{(n)}A^{(n)}(R^{(n)})^{-1}$, which implies that the dToda equation yields the matrix transformations from $A^{(n)}$ to $A^{(n+1)}$ without changing the eigenvalues. Thus, Lax representations are a good starting point for understanding other soliton BBSs.

Recapturing the Lax representation (3.3) in terms of a numerical technique in linear algebra can be regarded as an LR transformation [24], which first decomposes $A^{(n)}$ into a product of lower and upper bidiagonal matrices and then generates $A^{(n+1)}$ by inverting the product. The dToda equation can thus generate LR transformations of tridiagonal matrices. In fact, the dToda equation (3.2) is the recursion formula of the well-known quotient-difference (qd) algorithm [24], which computes symmetric tridiagonal eigenvalues. The qd algorithm was originally designed based on a sequence of tridiagonal LR transformations.

The discrete Lotka–Volterra (dLV) system [33] also has a close relationship with tridiagonal LR transformations. This is consistent with the dToda equation and dLV system being linked by a variable transformation called the Bäcklund transformation (for the dToda-dLV case, this is specifically the Miura transformation). The dLV system is a time-discretization of the predator-prey Lotka–Volterra (LV) system, and is expressed using the time-discretization parameter $\delta^{(n)}$ as follows:

$$\begin{cases} u_k^{(n+1)}(1+\delta^{(n+1)}u_{k-1}^{(n+1)}) = u_k^{(n)}(1+\delta^{(n)}u_{k+1}^{(n)}), \quad k=1,2,\ldots,2m-1, \\ u_0^{(n)} := 0, \quad u_{2m}^{(n)} := 0, \end{cases} \tag{3.5}$$

where $u_k^{(n)}$ corresponds to the number of members of the kth species at discrete-time n. For the kth species, the dLV system (3.5) assumes that the $(k-1)$th species is the predator and the $(k+1)$th species is the prey. A time-delay version of the dLV system (3.5) has been formulated by enforcing a time delay before preying and being preyed on; this is related to multifold LR transformations of tridiagonal matrices

[25]. Similar to dToda, the dLV and delayed dLV systems can be applied to compute the singular values of bidiagonal matrices, which is the mathematical equivalent of computing the eigenvalues of symmetric tridiagonal matrices. Here, we emphasize that the LR transformation perspective transforms nonlinear phenomena into linear algebra.

We now return to the topic of the BBSs. An extension of a simple BBS is a BBS with numbered balls, where each ball performs a more complicated motion than in the simple BBS [32]. This is called a numbered BBS (nBBS) and differs from the simple BBS because it is not designed based on the time-discretization of an integrable system and its ultradiscretization. The discrete integrable system associated with the nBBS was actually derived from the inverse ultradiscretization of its equation of motion. Considering the hungry extensions of the LV system [2, 14, 30] and its discretization, the resulting equation was named the discrete hungry Toda (dhToda) equation. Hungry LV (hLV) systems, sometimes referred to as Bogoyavlenskii lattices, describe multiple predator-prey interactions, which describes one species preying on multiple species because it is hungrier.

The LR transformation perspective plays a key role in determining the Bäcklund transformation between the dhToda equation and one discrete hLV (dhLV) system, as does formulating another type of dhToda equation that is different from the nBBS-origin equation and is linked to the other dhLV system. In addition, the LR transformation perspective is useful to clarify continuous-time analogues of dhToda equations [27]. Note that the Lax representation is equivalent to the related LR transformation for dToda, but not for dhTodas. With either approach, the dToda equation is translated into a single-matrix equation. In contrast, Lax representations of the dhToda equations are a single-matrix equation but LR transformations related to the dhToda equations are expressed using a multiple-matrix equation. Moreover, by combining the LR transformations with another matrix transformation, we can naturally derive nonautonomous versions of the dhToda equations, which are the dhToda equations with an arbitrary parameter.

The remainder of this chapter is organized as follows. In Sect. 3.2, we describe discrete hungry integrable systems, which are dhToda equations, their nonautonomous versions, and dhLV systems, using the framework of LR transformation. We also describe the relationship between discrete hungry integrable systems and computing matrix eigenvalues. In Sect. 3.3, we relate the discrete relativistic Toda (drToda) equation to LR transformations, and clarify the link between the drToda equation and dhLV system by taking advantage of the flexibility of the LR transformation perspective. In Sect. 3.4, by considering ultradiscrete analogues of the L and R factors in LR transformations, we demonstrate that the udToda and udhToda equations generate matrix transformations that preserve an eigenvalue over min-plus algebra and can be used to compute the eigenvalues of the min-plus matrices. These properties correspond to the simple BBS and nBBS having conserved quantities under discrete-time evolution. In Sect. 3.5, we present an nBBS based on a dhToda equation derived from the LR perspective. Finally, in Sect. 3.6, we give concluding remarks.

3.2 Discrete Hungry Integrable Systems

In this section, we first relate two types of dhToda equations to the LR transformations of Hessenberg matrices. We then derive nonautonomous extensions of the dhToda equations by introducing shifts in the LR transformations and show their relationship to dhLV systems. We also describe the asymptotic convergence of discrete hungry integrable systems as discrete-time tends to infinity.

3.2.1 Discrete Hungry Toda Equations

An extension of the dToda equation, the dhToda equation [32] is written as

$$\begin{cases} q_k^{(n+M)} + e_{k-1}^{(n+1)} = q_k^{(n)} + e_k^{(n)}, \quad k = 1, 2, \dots, m, \\ q_k^{(n+M)} e_k^{(n+1)} = q_{k+1}^{(n)} e_k^{(n)}, \quad k = 1, 2, \dots, m-1, \\ e_0^{(n)} := 0, \quad e_m^{(n)} := 0, \quad n = 0, 1, \dots, \end{cases} \tag{3.6}$$

where M is a positive integer. We refer to (3.6) as the dhToda-I equation to distinguish it from other dhToda equations that will appear later. Note that the dhToda-I equation with $M = 1$ coincides with the dToda equation (3.2). In general, the values of q and e are uniquely determined at every discrete-time n if $\{q_k^{(0)}\}_{k=1}^m$, $\{q_k^{(1)}\}_{k=1}^m, \dots, \{q_k^{(M-1)}\}_{k=1}^m$ and $\{e_k^{(0)}\}_{k=1}^{m-1}$ are given. Using (3.4), we obtain the Lax representation of the dhToda-I equation:

$$L^{(n+1)} R^{(n+M)} = R^{(n)} L^{(n)}, \quad n = 0, 1, \dots. \tag{3.7}$$

By applying the discrete-time evolutions generated by (3.7) to the upper Hessenberg matrix $A^{(n)} := L^{(n)} R^{(n+M-1)} R^{(n+M-2)} \cdots R^{(n)}$, we derive

$$A^{(n+1)} = R^{(n)} A^{(n)} (R^{(n)})^{-1}, \quad n = 0, 1, \dots. \tag{3.8}$$

This implies that the dhToda-I equation generates $A^{(1)}, A^{(2)}, \dots$ with the same eigenvalues as those of $A^{(0)}$. We can rewrite the similarity transformation (3.8) as

$$\begin{cases} A^{(n)} = L^{(n)} (R^{(n+M-1)} R^{(n+M-2)} \cdots R^{(n)}), \\ (R^{(n+M-1)} R^{(n+M-2)} \cdots R^{(n)}) L^{(n)} = A^{(n+M)}. \end{cases} \tag{3.9}$$

Thus, the dhToda-I equation first decomposes $A^{(n)}$ into the product of a lower bidiagonal L and multiple upper bidiagonal R factors, and then constructs $A^{(n+M)}$ by swapping the L and R factors M times. Moreover, by observing the entries of $A^{(n)}$, the asymptotic convergence as $n \to \infty$ of this sequence of LR transformations can be used to compute the eigenvalues of $A^{(0)}$.

Theorem 3.1 (Fukuda et al. [7]) *Suppose that $A^{(0)}$ is totally nonnegative (TN), that is, $q_k^{(j)} > 0$ for $j = 0, 1, \ldots, M-1$ and $e_k^{(0)} > 0$. Then, $A^{(n)}$ converges to an upper triangular matrix as $n \to \infty$ and*

$$\lim_{n\to\infty} q_k^{(n)} q_k^{(n-1)} \cdots q_k^{(n-M+1)} = \lambda_k(A^{(0)}), \quad k = 1, 2, \ldots, m,$$
$$\lim_{n\to\infty} e_k^{(n)} = 0, \quad k = 1, 2, \ldots, m-1,$$

where $\lambda_k(A^{(0)})$ denotes the kth largest eigenvalue of $A^{(0)}$.

Another hungry extension of the dToda equation is the dhToda-II equation [21] and is expressed as

$$\begin{cases} q_k^{(n+1)} + e_{k-1}^{(n+M)} = q_k^{(n)} + e_k^{(n)}, & k = 1, 2, \ldots, m, \\ q_k^{(n+1)} e_k^{(n+M)} = q_{k+1}^{(n)} e_k^{(n)}, & k = 1, 2, \ldots, m-1, \\ e_0^{(n)} := 0, \quad e_m^{(n)} := 0, & n = 0, 1, \ldots. \end{cases} \tag{3.10}$$

As for the dhToda-I equation, the dhToda-II equation with $M = 1$ is the dToda equation (3.2). However, the dhToda-II equation differs from the dhToda-I equation in that the initial values required to uniquely determine all q and e are $\{q_k^{(0)}\}_{k=1}^m$ and $\{e_k^{(0)}\}_{k=1}^{m-1}, \{e_k^{(1)}\}_{k=1}^{m-1}, \ldots, \{e_k^{(M-1)}\}_{k=1}^{m-1}$ rather than $\{q_k^{(0)}\}_{k=1}^m, \{q_k^{(1)}\}_{k=1}^m, \ldots, \{q_k^{(M-1)}\}_{k=1}^m$ and $\{e_k^{(0)}\}_{k=1}^{m-1}$. Additionally, in the Lax representation of the dhToda-II equation, an arbitrary M appears in the superscript of the L factor instead of the R factor, as follows:

$$L^{(n+M)} R^{(n+1)} = R^{(n)} L^{(n)}, \quad n = 0, 1, \ldots. \tag{3.11}$$

Preparing the lower Hessenberg matrix $A^{(n)} := L^{(n)} L^{(n+1)} \cdots L^{(n+M-1)} R^{(n)}$ shows that the dhToda-II equation yields a sequence of matrices with the same eigenvalues as those of $A^{(0)}$. This is because (3.11) leads to $A^{(n+1)} = (L^{(n)})^{-1} A^{(n)} L^{(n)}$ for $n = 0, 1, \ldots$. Moreover, (3.11) leads to

$$\begin{cases} A^{(n)} = (L^{(n)} L^{(n+1)} \cdots L^{(n+M-1)}) R^{(n)}, \\ R^{(n)} (L^{(n)} L^{(n+1)} \cdots L^{(n+M-1)}) = A^{(n+M)}, \end{cases}$$

where the number of L and R factors is reversed compared with in (3.9). Here, we emphasize that (3.10) was originally formulated from an LR transformation rather than realistic dynamics. As $n \to \infty$, the dhToda-II equation exhibits asymptotic convergence similar to that described in Theorem 3.1.

Theorem 3.2 (Nishiyama et al. [21]) *Suppose that $A^{(0)}$ is TN, that is, $q_k^{(0)} > 0$ and $e_k^{(j)} > 0$ for $j = 0, 1, \ldots, M-1$. Then, $A^{(n)}$ converges to a lower triangular matrix as $n \to \infty$ and*

$$\lim_{n\to\infty} q_k^{(n)} = \lambda_k(A^{(0)}), \quad k = 1, 2, \ldots, m,$$
$$\lim_{n\to\infty} e_k^{(n)} = 0, \quad k = 1x, 2, \ldots, m-1.$$

3.2.2 Nonautonomous Discrete Hungry Integrable Systems

We now consider the LR transformations of upper Hessenberg matrices. Introducing shifts into the LR transformations is useful for accelerating convergence. A shifted version of (3.9) is given by

$$\begin{cases} A^{(n)} - s^{(n)} I = \hat{L}_0^{(n)} \hat{R}^{(n)}, \\ \hat{R}^{(n)} \hat{L}_0^{(n)} + s^{(n)} I = A^{(n+M)}, \end{cases} \tag{3.12}$$

where $\hat{L}_0^{(n)}$ is a lower bidiagonal matrix, $\hat{R}^{(n)}$ is an upper triangle matrix, and $s^{(n)}$ denotes the shift parameter. Since $A^{(n+M)} = (\hat{L}_0^{(n)})^{-1} A^{(n)} \hat{L}_0^{(n)}$ in (3.12), the eigenvalues of $A^{(n+M)}$ are identical to those of $A^{(n)}$. Such a shift, which does not change the eigenvalues, is called an "implicit" shift. In contrast, an "explicit" shift changes the eigenvalues. We can verify that the implicit-shift LR transformation (3.12) is generated by a pair of LR and LL transformations, such that

$$\begin{cases} \hat{L}_{\ell+1}^{(n)} R^{(n+M+\ell)} = R^{(n+\ell)} \hat{L}_\ell^{(n)}, \quad \ell = 0, 1, \ldots, M-1, \\ \hat{L}_0^{(n,0)} L^{(n+M)} = L^{(n)} \hat{L}_M^{(n)}, \end{cases} \tag{3.13}$$

where $\hat{L}_\ell^{(n)}$ are bidiagonal matrices involving new variables $\hat{e}_{\ell,k}^{(n)}$ and are given by

$$\hat{L}_\ell^{(n)} = \begin{bmatrix} 1 & & & \\ \hat{e}_{\ell,1}^{(n)} & 1 & & \\ & \ddots & \ddots & \\ & & \hat{e}_{\ell,m-1}^{(n)} & 1 \end{bmatrix}.$$

Using the entry identities and boundary conditions in (3.13), we can derive a recursion formula involving an arbitrary $s^{(n)}$:

$$\begin{cases} q_k^{(n+M+\ell)} + \hat{e}_{\ell+1,k-1}^{(n)} = q_k^{(n+\ell)} + \hat{e}_{\ell,k}^{(n)}, \quad k = 1, 2, \ldots, m, \quad \ell = 0, 1, \ldots, M-1, \\ q_k^{(n+M+\ell)} \hat{e}_{\ell+1,k}^{(n)} = q_{k+1}^{(n+\ell)} \hat{e}_{\ell,k}^{(n)}, \quad k = 1, 2, \ldots, m, \quad \ell = 0, 1, \ldots, M-1, \\ e_k^{(n+M)} + \hat{e}_{0,k}^{(n)} = e_k^{(n)} + \hat{e}_{M,k}^{(n)}, \quad k = 1, 2, \ldots, m-1, \\ \hat{e}_{0,1}^{(n)} = \dfrac{q_1^{(n)} q_1^{(n+1)} \cdots q_1^{(n+M-1)}}{q_1^{(n)} q_1^{(n+1)} \cdots q_1^{(n+M-1)} - s^{(n)}} e_1^{(n)}, \\ e_k^{(n+M)} \hat{e}_{0,k+1}^{(n)} = e_{k+1}^{(n)} \hat{e}_{M,k}^{(n)}, \quad k = 1, 2, \ldots, m-1, \\ \hat{e}_{\ell,0}^{(n)} := 0, \quad \hat{e}_{\ell,m}^{(n)} := 0, \quad \ell = 0, 1, \ldots, M-1, \\ e_0^{(n)} := 0, \quad e_m^{(n)} := 0. \end{cases} \tag{3.14}$$

Since (3.14) with $s^{(n)} = 0$ simplifies to the dhToda-I equation, we can regard (3.14) as a nonautonomous dhToda-I equation with an arbitrary parameter. As $n \to \infty$, the nonautonomous dhToda-I equation exhibits the same asymptotic convergence to the upper Hessenberg eigenvalues as (3.6).

Theorem 3.3 (Fukuda et al. [10]) *Suppose that $A^{(0)}$ is TN, that is, $q_k^{(0)} > 0$ and $e_k^{(j)} > 0$ for $j = 0, 1, \ldots, M-1$. If $s^{(nM)} < \lambda_m(A^{(0)})$ for $n = 0, 1, \ldots,$ then the following holds:*

$$\lim_{\ell \to \infty} q_k^{(\ell M)} q_k^{(\ell M+1)} \cdots q_k^{(\ell M+M-1)} = \lambda_k(A^{(0)}), \quad k = 1, 2, \ldots, m,$$
$$\lim_{\ell \to \infty} e_k^{(\ell M)} = 0, \quad k = 1, 2, \ldots, , m-1.$$

The shift strategy and its effect on the computation of eigenvalues can be found in [11].

Similar to the dhToda-I equation, considering the implicit-shift LR transformation of the lower Hessenberg matrices yields the nonautonomous dhToda-II equation with an arbitrary $s^{(n)}$ [28]:

$$\begin{cases} e_{\ell,k-1}^{(n+1)} + \hat{q}_{\ell+1,k}^{(n)} = e_{\ell,k}^{(n)} + \hat{q}_{\ell,k}^{(n)}, \quad k = 1, 2, \ldots, m, \quad \ell = 0, 1, \ldots, M-1, \\ e_{\ell,k}^{(n+1)} \hat{q}_{\ell+1,k}^{(n)} = e_{\ell,k}^{(n)} \hat{q}_{\ell,k+1}^{(n)}, \quad k = 1, 2, \ldots, m-1, \quad \ell = 0, 1, \ldots, M-1, \\ q_k^{(n+1)} + \hat{q}_{0,k+1}^{(n)} = q_{k+1}^{(n)} + \hat{q}_{M,k}^{(n)}, \quad k = 1, 2, \ldots, m, \\ \hat{q}_{0,1}^{(n)} = q_1^{(n)} - s^{(n)}, \\ q_k^{(n+1)} \hat{q}_{0,k}^{(n)} = q_k^{(n)} \hat{q}_{M,k}^{(n)}, \quad k = 1, 2, \ldots, m, \\ e_{\ell,0}^{(n)} := 0, \quad e_{\ell,m}^{(n)} := 0. \end{cases} \tag{3.15}$$

We can rewrite the discrete-time evolutions generated by (3.15) using the LR and RR transformations as

$$\begin{cases} L_\ell^{(n+1)}\hat{R}_{\ell+1}^{(n)} = \hat{R}_\ell^{(n)}L_\ell^{(n)}, \quad \ell = 0, 1, \ldots, M-1, \\ R^{(n+1)}\hat{R}_0^{(n)} = \hat{R}_M^{(n)}R^{(n)}, \end{cases} \tag{3.16}$$

$$L_\ell^{(n)} := \begin{bmatrix} 1 & & & \\ e_{\ell,1}^{(n)} & 1 & & \\ & \ddots & \ddots & \\ & & e_{\ell,m-1}^{(n)} & 1 \end{bmatrix}, \quad \hat{R}_\ell^{(n)} := \begin{bmatrix} \hat{q}_{\ell,1}^{(n)} & 1 & & \\ & \hat{q}_{\ell,2}^{(n)} & \ddots & \\ & & \ddots & 1 \\ & & & \hat{q}_{\ell,m}^{(n)} \end{bmatrix}.$$

Moreover, a combination of LR and RR transformations leads to the implicit-shift lower Hessenberg LR transformation from $A^{(n)}$ to $A^{(n+1)}$:

$$\begin{cases} A^{(n)} - s^{(n)}I = \hat{L}^{(n)}\hat{R}_0^{(n)}, \\ \hat{R}_0^{(n)}\hat{L}^{(n)} + s^{(n)}I = A^{(n+M)}, \end{cases}$$

where $\hat{L}^{(n)}$ is a lower triangular matrix and $\hat{R}_0^{(n)}$ is an upper bidiagonal matrix. Expressing the solution to the nonautonomous dhToda-II equation using determinants, and examining their asymptotic expansions as $n \to \infty$, we can clarify the asymptotic convergence for the nonautonomous dhToda-II equation as $n \to \infty$ without restricting $A^{(0)}$ to being TN.

Theorem 3.4 (Shinjo et al. [28]) *If* $q_k^{(0)} \neq 0$ *and* $e_{\ell,k}^{(0)} \neq 0$ *for* $\ell = 0, 1, \ldots, M-1$, *then*

$$\lim_{n\to\infty} q_k^{(n)} = \lambda_k(A^{(0)}), \quad k = 1, 2, \ldots, m,$$
$$\lim_{n\to\infty} e_{\ell,k}^{(n)} = 0, \quad k = 1, 2, \ldots, m-1, \quad \ell = 0, 1, \ldots, M-1,$$
$$\lim_{n\to\infty} \hat{q}_{\ell,k}^{(n)} = \lambda_k(A^{(0)}) - s^*, \quad k = 1, 2, \ldots, m, \quad \ell = 0, 1, \ldots, M,$$

where s^* *is a constant such that* $s^{(n)} \to s^*$ *as* $n \to \infty$.

For the nonautonomous dhToda-I equation, we can also obtain a convergence theorem similar to Theorem 3.4 by analyzing the determinant solution.

We now introduce new variables $u_k^{(n)}$ that comprise the nonautonomous dhToda-II variables:

$$\begin{cases} q_k^{(n)} = u_{(M+1)(k-1)+1}^{(n)} \prod_{j=1}^{M}(1 + \delta^{(n)}u_{(M+1)(k-1)+1-j}^{(n)}), \\ e_{\ell-1,k}^{(n)} = u_{(M+1)(k-1)+\ell+1}^{(n)} \prod_{j=1}^{M}(1 + \delta^{(n)}u_{(M+1)(k-1)+j+1-j}^{(n)}), \\ \hat{q}_{\ell,k}^{(n)} = \dfrac{1}{\delta^{(n)}} \prod_{j=0}^{M}(1 + \delta^{(n)}u_{(M+1)(k-1)+j+1-j}^{(n)}). \end{cases} \tag{3.17}$$

Applying (3.17) to (3.15), we derive a dhLV system:

$$\begin{cases} u_k^{(n+1)}\displaystyle\prod_{j=1}^{M}(1+\delta^{(n+1)}u_{k-j}^{(n+1)}) = u_k^{(n)}\prod_{j=1}^{M}(1+\delta^{(n)}u_{k+j}^{(n)}), \\ k = 1, 2, \ldots, (M+1)m - M, \\ u_{\ell-M}^{(n)} := 0, \quad u_{(M+1)m+\ell-M}^{(n)} := 0, \quad \ell = 1, 2, \ldots, M, \quad n = 0, 1, \ldots. \end{cases} \tag{3.18}$$

Equation (3.18) is the Bäcklund transformation between the nonautonomous dhToda-II equation and dhLV system. We can systematically find this Bäcklund transformation from the perspective of the LR transformation. Details of Bäcklund transformations between the nonautonomous dhToda-I equation and dhLV systems can be found in [9, 26]. Combining these Bäcklund transformations with Theorems 3.3 and 3.4, we can also obtain convergence theorems for dhLV systems.

3.3 Discrete Relativistic Toda Equation

The relativistic Toda equation is a Poincaré-invariant generalization of the Toda equation. Since its introduction by Ruijsenaars [23], it has received considerable attention, and various studies have been conducted on its integrability [23], special solutions [3, 22], integrable discretization [29] and its solutions [18, 19], Lax representations [29], and application to numerical algorithms [19]. In this section, we consider the following nonautonomous discrete relativistic Toda (drToda) equation [20, 36]:

$$\begin{cases} a_k^{(n+1)} + s^{(n+1)}(1+b_k^{(n+1)}) = a_k^{(n)}\dfrac{1+b_{k+1}^{(n)}}{1+b_k^{(n)}} + s^{(n)}(1+b_{k+1}^{(n)}), \quad k = 0, 1, \ldots, K-1, \\ a_{k-1}^{(n+1)}b_k^{(n+1)} = a_k^{(n)}b_k^{(n)}\dfrac{1+b_{k+1}^{(n)}}{1+b_k^{(n)}}, \quad k = 1, 2, \ldots, K-1, \\ b_0^{(n)} := 0, \quad n = 0, 1, \ldots, \end{cases} \tag{3.19}$$

where $a_k^{(n)}$ and $b_k^{(n)}$ are variables at the kth grid point at discrete-time n and $s^{(n)}$ is a scalar parameter that can be chosen arbitrarily for each n. We consider this equation from the perspective of a shifted LR transformation so that we can discuss its derivation, Lax representation, relationship with other discrete integrable systems, and conserved quantities in an elementary and unified manner.

3.3.1 Derivation from the Perspective of a Shifted LR Transformation

We derive the nonautonomous drToda equation from the autonomous drToda equation, which is given by

$$\begin{cases} u_k^{(n)} + v_k^{(n+1)} = u_k^{(n+1)} + v_{k+1}^{(n)}, \quad k = 0, 1, \ldots, K-1, \\ u_k^{(n+1)} v_k^{(n)} = u_{k-1}^{(n)} v_k^{(n+1)}, \quad k = 1, 2, \ldots, K-1, \\ v_0^{(n)} := 0, \quad n = 0, 1, \ldots. \end{cases}$$

This resembles the discrete Toda equation, except for the differences in the indices. Its Lax representation is given by

$$R^{(n)}(L^{(n)})^{-1} = (L^{(n+1)})^{-1} R^{(n+1)}, \quad n = 0, 1, \ldots, \tag{3.20}$$

where

$$R^{(n)} = \begin{bmatrix} u_0^{(n)} & 1 & & \\ & u_1^{(n)} & \ddots & \\ & & \ddots & 1 \\ & & & u_{K-1}^{(n)} \end{bmatrix}, \quad L^{(n)} = \begin{bmatrix} 1 & & & \\ v_1^{(n)} & 1 & & \\ & \ddots & \ddots & \\ & & v_{K-1}^{(n)} & 1 \end{bmatrix}. \tag{3.21}$$

Equation (3.20) can be viewed as an LR transformation of the structured matrix $A^{(n)} := \left(L^{(n)}\right)^{-1} R^{(n)}$.

We now introduce a shift into the LR transformation (3.20). To simplify the notation, we denote the variables at discrete-time n without an asterisk, and those at discrete-time $n+1$ with an asterisk. The shifted LR transformation can then be written as follows:

$$A - sI = \breve{L}\breve{R}, \tag{3.22}$$

$$\breve{R}\breve{L} + sI = A^*. \tag{3.23}$$

Here, (3.22) decomposes the shifted matrix $A - sI$ into a product of L and R factors, and (3.23) forms the matrix A^* at the next time step by reversing the order of the product and re-adding the shift. Interestingly, we can show that if the matrix A can be written as $A = L^{-1}R$, then A^* can also be written as $A^* = (L^*)^{-1}R^*$, where L^* and R^* are of the form of (3.21). Furthermore, the structures of $\breve{L}$ and $\breve{R}$ can be identified as follows.

Lemma 3.1 (Yamamoto et al. [36]) *Let $L = I + L'$ and $R = D + J$, where J is an upper bidiagonal matrix with zeros on the diagonal and ones on the upper subdiagonal. Then, the matrices $\breve{L}$ and $\breve{R}$ defined by* (3.22) *with $A = L^{-1}R$ can be expressed as*

$$\breve{R} = \breve{D} + J, \tag{3.24}$$

$$\breve{L} = (I + L')^{-1}(I - sL'\breve{D}^{-1}), \tag{3.25}$$

where $\breve{D}$ *is the solution to the matrix equation:*

$$\breve{D} = D - s(I - L'\breve{D}^{-1}J). \tag{3.26}$$

Now, we can write the two steps of the shifted LR transformation (3.22) and (3.23) as

$$L^{-1}R - sI = \breve{L}\breve{R}, \tag{3.27}$$

$$\breve{R}\breve{L} + sI = (L^*)^{-1}R^*, \tag{3.28}$$

$$(L^*)^{-1}R^* - s^*I = \breve{L}^*\breve{R}^*, \tag{3.29}$$

$$\breve{R}^*\breve{L}^* + s^*I = (L^{**})^{-1}R^{**}, \tag{3.30}$$

where the symbols with double asterisks denote the variables at discrete-time $n + 2$. Then, we can regard $\breve{L}$ and $\breve{R}$ as basic variables and consider (3.28) and (3.29) as the equations that govern their discrete-time evolutions. Equations (3.28) and (3.29) can be rewritten as

$$\breve{R}\breve{L} - (s^* - s)I = \breve{L}^*\breve{R}^*. \tag{3.31}$$

This can be viewed as an explicitly shifted LR transformation with the shift $s^* - s$. By changing the variables from L' to $\tilde{L}' := -L'\breve{D}^{-1}$, we can rewrite (3.25) as

$$\breve{L} = (I - \tilde{L}'\breve{D})^{-1}(I + s\tilde{L}'). \tag{3.32}$$

By substituting (3.24) and (3.32) into (3.31), we can express the explicitly shifted LR transformation in terms of $\breve{D}$ and $\tilde{L}'$:

$$\begin{aligned}&(\breve{D} + J)(I - \tilde{L}'\breve{D})^{-1}(I + s\tilde{L}') + sI\\ &\quad = (I - \tilde{L}^{*\prime}\breve{D}^*)^{-1}(I + s^*\tilde{L}^{*\prime})(\breve{D}^* + J) + s^*I.\end{aligned} \tag{3.33}$$

Although this appears to be a complicated matrix equality, we can show that if the equalities between the diagonal entries and between the lower sub-diagonal entries hold, then equality automatically holds for the entire matrix. This leads to the following lemma.

Lemma 3.2 (Yamamoto et al. [36]) *Equation* (3.33) *is a combination of the following two equations:*

$$(I + J\tilde{L}')(\breve{D} + sI) = (\breve{D}^* + s^*I) + \tilde{L}^{*\prime}(\breve{D}^* + s^*I)J, \tag{3.34}$$
$$(I + J\tilde{L}')(\breve{D}\tilde{L}') = (\tilde{L}^{*\prime}\breve{D}^*)(I + J\tilde{L}'). \tag{3.35}$$

Here, (3.34) and (3.35) represent the equality between the diagonal entries and between the lower sub-diagonal entries, respectively.

To rewrite (3.34) and (3.35) entry-by-entry, let

$$\breve{D} = \begin{bmatrix} a_0 & & & \\ & a_1 & & \\ & & \ddots & \\ & & & a_{K-1} \end{bmatrix}, \quad \tilde{L}' = \begin{bmatrix} 0 & & & \\ b_1 & 0 & & \\ & \ddots & \ddots & \\ & & b_{K-1} & 0 \end{bmatrix},$$
$$b_0 = b_K = 0, \tag{3.36}$$

and assume the same expressions for the variables with asterisks. Then, (3.34) and (3.35) can be rewritten as

$$(1 + b_{k+1})(a_k + s) = (a_k^* + s^*) + b_k^*(a_{k-1}^* + s^*), \quad k = 0, 1, \ldots, K-1, \tag{3.37}$$
$$(1 + b_{k+1})a_k b_k = a_{k-1}^* b_k^*(1 + b_k), \quad k = 1, 2, \ldots, K-1. \tag{3.38}$$

Equation (3.38) is the second equation of (3.19). Rewriting (3.37) and using (3.38), we obtain

$$\begin{aligned} a_k^* + s^*(1 + b_k^*) &= a_k(1 + b_{k+1}) - a_{k-1}^* b_k^* + s(1 + b_{k+1}) \\ &= a_k(1 + b_{k+1}) - a_k b_k \frac{1 + b_{k+1}}{1 + b_k} + s(1 + b_{k+1}) \\ &= a_k \frac{1 + b_{k+1}}{1 + b_k} + s(1 + b_{k+1}), \quad k = 0, 1, \ldots, K-1, \end{aligned} \tag{3.39}$$

which is the first equation of (3.19). We summarize this observation as the main theorem of this section.

Theorem 3.5 ([36]) *One step of the nonautonomous drToda equation is equivalent to the explicitly shifted LR transformation* (3.31)*, where $\breve{L}$ and $\breve{R}$ are given by* (3.32) *and* (3.24)*, respectively, and $\breve{D}$ and $\tilde{L}'$ are given by* (3.36)*.*

This theorem enables us to derive a condition on s to ensure the absence of a breakdown in the discrete-time evolution and the positivity of the variables. Further details can be found in [36].

3.3.2 *Relationship with the Discrete Hungry Lotka–Volterra System*

From the discussion in Sect. 3.3.1, we know that the nonautonomous drToda equation can be expressed by two equivalent systems. The first is to use $\breve{L}$ and $\breve{R}$ (or $\breve{D}$ and $\tilde{L}'$) as the basic variables and (3.28) and (3.29) (or (3.38) and (3.39)) as the equations for discrete-time evolution. The second is to use L and R as the basic variables and use (3.27) and (3.28) for the discrete-time evolution. These two systems can be interchanged using (3.27) as the change-of-variables formula.

Using this fact, we can construct a Bäcklund transformation between the drToda equation (3.19) and dhLV system (3.18). To illustrate this, note that the dhLV system can be expressed as the following implicitly shifted LR transformation [6]:

$$\begin{cases} L_1 L_2 \cdots L_M R - sI = \breve{L}\breve{R}, \\ \breve{R}\breve{L} + sI = L_1^* L_2^* \cdots L_M^* R^*, \end{cases}$$

where L_i for $i = 1, 2, \ldots, M$ and R are given by

$$L_i = \begin{bmatrix} 1 & & & \\ e_{i,1} & 1 & & \\ & \ddots & \ddots & \\ & & e_{i,m-1} & 1 \end{bmatrix}, \quad R = \begin{bmatrix} q_1 & 1 & & \\ & q_2 & \ddots & \\ & & \ddots & 1 \\ & & & q_m \end{bmatrix}.$$

Consider the special case where $m = K$, $M = K - 1$, and only one lower subdiagonal entry, $e_{i,K-i}$, is nonzero in each L_i. Thus, we have

$$L_1 L_2 \cdots L_M = \begin{bmatrix} 1 & & & & \\ -e_{K-1,1} & 1 & & & \\ & \ddots & \ddots & & \\ & & -e_{2,K-2} & 1 & \\ & & & -e_{1,K-1} & 1 \end{bmatrix}^{-1}.$$

For this particular case, the dhLV system has the same form as that of (3.27) and (3.28). The relationship between the dhLV and drToda variables (expressed in terms of L and R) is

$$q_k = u_{k-1}, \quad k = 1, 2, \ldots, K, \tag{3.40}$$

$$e_{K-k,k} = -v_k, \quad k = 1, 2, \ldots, K - 1. \tag{3.41}$$

The dhLV variables can also be related to the drToda variables expressed in terms of $\breve{D}$ and $\tilde{L}'$. To this end, we note the following relationship between the drToda variables:

$$u_k = a_k + s(1 + b_k), \quad n = 0, 1, \ldots, K - 1, \tag{3.42}$$

$$v_k = -a_{k-1}b_k, \quad k = 1, 2, \ldots, K - 1. \tag{3.43}$$

Equation (3.42) is an entry-wise representation of (3.26), while (3.43) follows from $\tilde{L}' := -L'\breve{D}^{-1}$. Substituting these equations into (3.40) and (3.41) yields the Bäcklund transformation between the dhLV variables and drToda variables in (3.19):

$$q_k = a_{k-1} + s(1 + b_{k-1}), \quad k = 1, 2, \ldots, K, \tag{3.44}$$

$$e_{K-k,k} = a_{k-1}b_k, \quad k = 1, 2, \ldots, K - 1, \tag{3.45}$$

$$e_{i,k} = 0, \quad i \neq K - k, \ k = 1, 2, \ldots, K - 1. \tag{3.46}$$

Finally, as an application of this Bäcklund transformation, we present the conserved quantities of the drToda equation. Because the conserved quantities of the dhLV system are known as functions of q and e variables [15], we only need to rewrite them in terms of the drToda variables using (3.45) and (3.46). The result is given by the following theorem.

Theorem 3.6 *Let $U_1, U_2, \ldots, U_{2K-1}$ be defined by*

$$\begin{cases} U_{2i-1} = a_{i-1} + s(1 + b_{i-1}), \quad i = 1, 2, \ldots, K, \\ U_{2i} = a_{i-1}b_i, \quad i = 1, 2, \ldots, K - 1, \end{cases}$$

and define the relational operator $\ll$ as

$$i \ll j \quad \Leftrightarrow \quad i + \mathrm{mod}(i + 1, 2) + 2 \leq j.$$

Then,

$$C_p = \sum_{1 \leq i_1 \ll i_2 \ll \cdots \ll i_p \leq 2K-1} U_{i_1} U_{i_2} \cdots U_{i_p}, \quad p = 1, 2, \ldots, K$$

are conserved quantities of the drToda equation (3.19).

Viewing a discrete integrable system from the perspective of a shifted LR transformation makes determining its relationship with other discrete integrable systems and exploiting the results obtained for the latter easier. The Bäcklund transformation and conserved quantities derived in this subsection illustrate the effectiveness of this approach.

3.4 Ultradiscrete Toda Equation

In this section, we demonstrate that the eigenvalues of matrices can be obtained using the discrete-time evolution of the udToda equation (3.1). The discussion here does not consider eigenvalues over ordinary linear algebra, but over min-plus algebra. In other words, we clarify a min-plus analogue of the qd algorithm for linear eigenvalue computation.

3.4.1 Min-Plus Algebra

Min-plus algebra is a commutative and idempotent semiring and is closely related to weighted directed graphs constructed with sets of nodes and directed edges with weights. Let $\mathbb{R}_{\min} := \mathbb{R} \cup \{+\infty\}$. For $a, b \in \mathbb{R}_{\min}$, min-plus algebra has two binary operations:

$$\begin{cases} a \oplus b = \min\{a, b\}, \\ a \otimes b = a + b. \end{cases}$$

Here, $\oplus$ and $\otimes$ are both associative and commutative; $\otimes$ is distributive with respect to $\oplus$; and $\varepsilon := +\infty$ and $e := 0$ are the identities with respect to $\oplus$ and $\otimes$, respectively. Moreover, let $\oslash$ be the inverse of $\otimes$ such that $a \otimes b \oslash b = a$.

The set of all m-by-n min-plus matrices is denoted as $\mathbb{R}_{\min}^{m\times n}$. For $A = (a_{ij})$, $B = (b_{ij}) \in \mathbb{R}_{\min}^{m\times m}$, the matrix sum $A \oplus B = ([A \oplus B]_{ij})$ and product $A \otimes B = ([A \otimes B]_{ij})$ are respectively given by

$$\begin{cases} [A \oplus B]_{ij} = a_{ij} \oplus b_{ij} = \min\{a_{ij}, b_{ij}\}, \\ [A \otimes B]_{ij} = \displaystyle\bigoplus_{\ell=1}^{m} (a_{i\ell} \otimes b_{\ell j}) = \min_{\ell=1,2,\ldots,m} \{a_{i\ell} + b_{\ell j}\}. \end{cases}$$

The following definition provides reasonable min-plus analogues for eigenvalues and eigenvectors.

Definition 3.1 For $A \in \mathbb{R}_{\min}^{m\times m}$, if $\lambda \in \mathbb{R}_{\min}$ and the vector $\boldsymbol{x} \in \mathbb{R}_{\min}^{m}$, which is not equal to $(\varepsilon, \varepsilon, \ldots, \varepsilon)^\top$, satisfy

$$A \otimes \boldsymbol{x} = \lambda \otimes \boldsymbol{x},$$

then λ and $\boldsymbol{x}$ are an eigenvalue of A and its corresponding eigenvector, respectively.

The following definition provides the min-plus analogue of determinants.

Definition 3.2 The tropical determinant $\mathrm{tropdet}(A)$ of $A \in \mathbb{R}_{\min}^{m\times m}$ is defined as

$$\mathrm{tropdet}(A) := \bigoplus_{\sigma \in S_m} a_{1\sigma(1)} \otimes a_{2\sigma(2)} \otimes \cdots \otimes a_{m\sigma(m)},$$

where S_m denotes the symmetric group of permutations of $\{1, 2, \ldots, m\}$. Moreover, the characteristic polynomial $f_A(\lambda)$ for $A \in \mathbb{R}_{\min}^{m\times m}$ is given by

$$f_A(\lambda) := \mathrm{tropdet}(A \oplus \lambda \otimes I),$$

where I denotes the m-by-m identity matrix, the (i, j) entry of which is 0 if $i = j$ or ε otherwise.

Proposition 3.1 (Maclagan-Sturmfels [17]) *The characteristic polynomial $f_A(\lambda)$ of $A \in \mathbb{R}_{\min}^{m\times m}$ is factorized as*

$$f_A(\lambda) \equiv (\lambda \oplus p_1)^{q_1} \otimes (\lambda \oplus p_2)^{q_2} \otimes \cdots \otimes (\lambda \oplus p_k)^{q_k},$$

where $p_1 < p_2 < \cdots < p_k$, $q_1 + q_2 + \cdots + q_k = m$, and the symbol "$\equiv$" indicates that the graphs of the functions are equal. The minimum root p_1 of $f_A(\lambda)$ coincides with the eigenvalue of A.

For a weighted digraph G involving m vertices, the entries of the m-by-m weighted adjacency matrix $A(G) = (a_{ij}) \in \mathbb{R}_{\min}^{m\times m}$ are given by

$$a_{ij} = \begin{cases} w(\mathrm{e}), & \text{if } \mathrm{e} = (\mathrm{v}_i, \mathrm{v}_j) \in E, \\ \varepsilon, & \text{otherwise.} \end{cases}$$

Conversely, for any $A \in \mathbb{R}_{\min}^{m\times m}$, there exists a weighted digraph $G(A)$ whose weighted adjacency matrix is A. In a circuit C on a weighted digraph $G(A)$, the number of edges and sum of edge weights are referred to as the length $\ell(C)$ and weight $w(C)$ of C, respectively. The average weight $w_{\mathrm{ave}}(C)$ is the ratio of the weight $w(C)$ to the length $\ell(C)$:

$$w_{\mathrm{ave}}(C) = \frac{w(C)}{\ell(C)}.$$

The following theorem provides an interesting relationship between the eigenvalue of $A \in \mathbb{R}_{\min}^{m\times m}$ and the average weight of the circuits on the weighted digraph $G(A)$.

Theorem 3.7 (Baccelli et al. [1]) *Let the weighted digraph $G(A)$ be strongly connected. Then, the weighted adjacency matrix $A(G)$ only has one eigenvalue. Moreover, the minimum value of the average weights of the circuits in $G(A)$ coincides with the eigenvalue of $A(G)$.*

3.4.2 Relationship with Min-Plus Eigenvalue

The ultradiscrete Toda (udToda) equation (3.1) is obtained by applying variable transformations $q_k^{(n)} = e^{-Q^{(n)}/\varepsilon}$ and $e_k^{(n)} = e^{-E^{(n)}/\varepsilon}$ to the dToda equation (3.2), taking the logarithm, multiplying both sides by ε, and taking the limit $\varepsilon \to +0$. Using the notation of min-plus algebra, that is, $\oplus$ and $\otimes$, the udToda equation can be rewritten as

$$\begin{cases} Q_k^{(n+1)} = \bigotimes_{j=1}^{k} Q_j^{(n)} \oslash \bigotimes_{j=1}^{k-1} Q_j^{(n+1)} \oplus E_k^{(n)}, \quad k = 1, 2, \ldots, m, \\ E_k^{(n+1)} = Q_{k+1}^{(n)} \otimes E_k^{(n)} \oslash Q_k^{(n+1)}, \quad k = 1, 2, \ldots, m-1, \\ E_0^{(n)} := \varepsilon, \quad E_m^{(n)} := \varepsilon, \quad n = 0, 1, \ldots. \end{cases}$$

Let $\mathcal{L}^{(n)}$ and $\mathcal{R}^{(n)}$ in $\mathbb{R}_{\min}^{m\times m}$ be ultradiscrete analogues of $(R^{(n)})^\top$ and $(L^{(n)})^\top$ in (3.3), respectively.

$$\mathcal{L}^{(n)} = \begin{bmatrix} Q_1^{(n)} & \varepsilon & \cdots\cdots & \varepsilon \\ e & Q_2^{(n)} & \ddots & \vdots \\ \varepsilon & \ddots & \ddots\ \ddots & \vdots \\ \vdots & \ddots & \ddots\ \ddots & \varepsilon \\ \varepsilon & \cdots & \varepsilon\ \ e & Q_m^{(n)} \end{bmatrix}, \quad \mathcal{R}^{(n)} = \begin{bmatrix} e & E_1^{(n)} & \varepsilon & \cdots & \varepsilon \\ \varepsilon & e & E_2^{(n)} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \varepsilon \\ \vdots & & \ddots & \ddots & E_{m-1}^{(n)} \\ \varepsilon & \cdots & \cdots & \varepsilon & e \end{bmatrix}. \tag{3.47}$$

Similar to the discrete case, we define the tridiagonal matrix $\mathcal{A}^{(n)} \in \mathbb{R}_{\min}^{m\times m}$ as the product of $\mathcal{L}^{(n)}$ and $\mathcal{R}^{(n)}$:

$$\begin{aligned} \mathcal{A}^{(n)} &:= \mathcal{L}^{(n)} \otimes \mathcal{R}^{(n)} \\ &= \begin{bmatrix} Q_1^{(n)} & Q_1^{(n)} \otimes E_1^{(n)} & \varepsilon & \cdots & \varepsilon \\ e & Q_2^{(n)} \oplus E_1^{(n)} & Q_2^{(n)} \otimes E_2^{(n)} & \ddots & \vdots \\ \varepsilon & \ddots & \ddots & \ddots & \varepsilon \\ \vdots & \ddots & \ddots & \ddots & Q_{m-1}^{(n)} \otimes E_{m-1}^{(n)} \\ \varepsilon & \cdots & \varepsilon & e & Q_m^{(n)} \oplus E_{m-1}^{(n)} \end{bmatrix}. \end{aligned}$$

To characterize the eigenvalues of $\mathcal{A}^{(n)}$ without the corresponding weighted digraph, we consider the characteristic polynomials of the min-plus tridiagonal matrices $\mathcal{A}^{(n)}$.

Theorem 3.8 (Watanabe et al. [34]) *The characteristic polynomials $f_{\mathcal{A}^{(n)}}(\lambda)$ of the tridiagonal matrices $\mathcal{A}^{(n)} \in \mathbb{R}_{\min}^{m\times m}$ can be factorized as*

$$f_{\mathcal{A}^{(n)}}(\lambda) = (\lambda \oplus Q_1^{(n)}) \otimes (\lambda \oplus Q_2^{(n)} \oplus E_1^{(n)}) \otimes \cdots \otimes (\lambda \oplus Q_m^{(n)} \oplus E_{m-1}^{(n)}).$$

Theorem 3.8 suggests that $Q_1^{(n)}, Q_2^{(n)} \oplus E_1^{(n)}, \ldots, Q_m^{(n)} \oplus E_{m-1}^{(n)}$ are the roots of the characteristic polynomial $f_{\mathcal{A}^{(n)}}(\lambda)$. From Proposition 3.1, the minimum root of $f_{\mathcal{A}^{(n)}}(\lambda)$ is an eigenvalue of $\mathcal{A}^{(n)}$. Thus, we can express the eigenvalues $\lambda^{(n)}$ using udToda variables:

$$\lambda^{(n)} = \bigoplus_{k=1}^{m} Q_k^{(n)} \oplus \bigoplus_{k=1}^{m-1} E_k^{(n)}. \tag{3.48}$$

The right-hand side of (3.48) coincides with the conserved quantities of the udToda equation [32]. This means that $\lambda^{(n)}$ is a constant, that is, it is independent of discrete-time n. Thus, the eigenvalue $\lambda = \lambda^{(0)}$ of the tridiagonal $\mathbb{R}_{\min}^{m\times m}$ matrix $\mathcal{A}^{(0)}$ is given by the right-hand side of (3.48) for any discrete-time n. In the BBS, the interchange of ball groups terminates at a sufficiently large discrete-time n. Simultaneously, the number of empty boxes between two neighboring ball groups monotonically increases as n increases. Recall that $Q_k^{(n)}$ corresponds to the number of successive balls in the kth ball group. A property of the values of $Q_k^{(n)}$ and $E_k^{(n)}$ [32] is that there exists a discrete-time N such that, for any $n \geq N$, the following holds:

$$Q_1^{(n)} \leq E_k^{(n)}, \quad k = 1, 2, \ldots, m-1, \tag{3.49}$$

$$Q_1^{(n)} \leq Q_2^{(n)} \leq \cdots \leq Q_m^{(n)}. \tag{3.50}$$

This implies that for a sufficiently large n, $Q_1^{(n)}$ is equal to the minimum value of all udToda variables. Combining (3.49) and (3.50) with (3.48) leads to the following theorem.

Theorem 3.9 (Watanabe et al. [34]) *For the eigenvalue λ of $\mathcal{A}^{(0)} \in \mathbb{R}_{\min}^{m\times m}$, the following holds for $n \geq N$:*

$$\lambda = Q_1^{(n)}.$$

Under the initial condition that $Q_1^{(0)}, Q_2^{(0)}, \ldots, Q_m^{(0)}$ and $E_1^{(0)}, E_2^{(0)}, \ldots, E_{m-1}^{(0)}$ are provided by the tridiagonal matrices $\mathcal{A}^{(0)} \in \mathbb{R}_{\min}^{m\times m}$, the udToda equation generates matrix transformations of $\mathcal{A}^{(0)}$ without changing an eigenvalue, and then makes the values of $Q_1^{(n)}$ equal to the eigenvalue after sufficient discrete-time evolution. Furthermore, in the weighted digraph $G(\mathcal{A}^{(0)})$, the value of $Q_1^{(n)}$ for a sufficiently large n is equal to the minimum value of the average weights of all circuits.

The ultradiscrete hungry Toda-I (udhToda-I) equation, which is a hungry extension of the ultradiscrete Toda equation, can also be related to eigenvalues over the min-plus algebra in a similar manner as for the udToda equation. The udhToda-I equation is obtained by the ultradiscretization of the dhToda-I equation (3.6) as follows:

$$\begin{cases} Q_k^{(n+M)} = \bigotimes_{j=1}^{k} Q_j^{(n)} \oslash \bigotimes_{j=1}^{k-1} Q_j^{(n+M)} \oplus E_k^{(n)}, \quad k = 1, 2, \ldots, m, \\ E_k^{(n+1)} = Q_{k+1}^{(n)} \otimes E_k^{(n)} \oslash Q_k^{(n+M)}, \quad k = 1, 2, \ldots, m-1, \\ E_0^{(n)} := \varepsilon, \quad E_m^{(n)} := \varepsilon, \quad n = 0, 1, \ldots. \end{cases} \tag{3.51}$$

We define the lower Hessenberg matrix $\mathcal{A}^{(n)} \in \mathbb{R}_{\min}^{m\times m}$ as the product of $\mathcal{L}^{(n)}$ and $\mathcal{R}^{(n)}$ in (3.47):

$$\mathcal{A}^{(n)} := \mathcal{L}^{(n)} \otimes \mathcal{L}^{(n+1)} \otimes \cdots \otimes \mathcal{L}^{(n+M-1)} \otimes \mathcal{R}^{(n)}.$$

There exist some circuits for the weighted digraph $G(\mathcal{A}^{(n)})$ for the min-plus matrix $\mathcal{A}^{(n)}$. The digraph is strongly connected; from Theorem 3.7, the eigenvalue of $\mathcal{A}^{(n)}$ coincides with the minimum average weight of all circuits of $G(\mathcal{A}^{(n)})$. Considering the udhToda-I equation, specifically the asymptotic behavior of the corresponding nBBS and its conserved quantities, we obtain the following theorem.

Theorem 3.10 (Kan et al. [16]) *For the eigenvalue λ of $\mathcal{A}^{(0)} \in \mathbb{R}_{\min}^{m\times m}$, the following holds for $n \geq N$:*

$$\lambda = Q_1^{(n)} \otimes Q_1^{(n+1)} \otimes \cdots \otimes Q_1^{(n+M-1)}.$$

Further details can be found in [16]. The ultradiscretization of the dLV system (3.5) can also be related to min-plus eigenvalues. Further details on this are provided in [8].

3.5 Numbered Box and Ball System

According to [32], the udhToda-I equation (3.51) governs the nBBS, where every ball is numbered. Based on the ultradiscretization of the dhToda-II equation (3.10), found in the study of Hessenberg LR transformations, we can design an nBBS where every box (rather than ball) is numbered [36]. To distinguish between the two types of nBBSs, we refer to the former and latter as nBBS-I and nBBS-II, respectively. In this section, we describe the numbering rules for the boxes, moving rules for the balls, and soliton behavior of the ball groups for the nBBS-II.

At discrete-time n, the nBBS-II with M types of boxes enforces that all boxes and balls satisfy the following rules:

(a) Only boxes between the leftmost and rightmost ball groups are numbered.
(b) Each box is numbered by one of the following integers: $n, n+1, \ldots, n+M-1$.
(c) Each box array, which is an array of successive empty boxes, contains at least one box with the number $n+i$ for $i = 0, 1, \ldots, M-1$.
(d) The boxes in each box array are lined up in the increasing order of their assigned numbers.

The discrete-time evolution from n to $n+1$ allows the identification numbers of the boxes to change. The balls move according to the following rules:

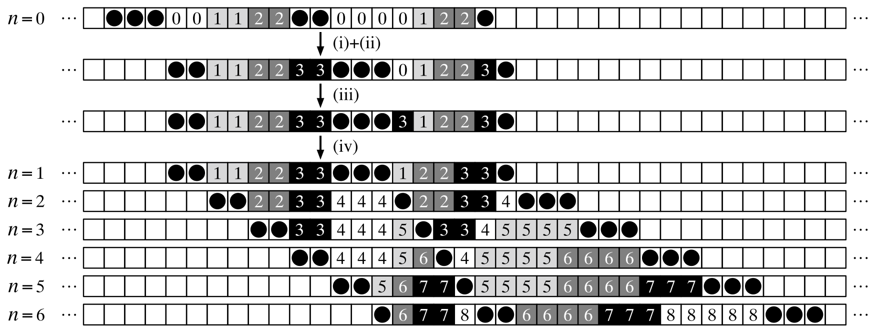


Fig. 3.2 An example of discrete-time evolution from $n = 0$ to $n = 6$ for the nBBS-II with $M = 3$

(i) Starting from the leftmost ball, each ball moves, one by one, to the nearest empty box to its right with the number n or without a number.

(ii) Remove the number n from the boxes that have been newly filled with a ball. Assign the number n to the boxes that have become empty, except for the boxes to the left of the leftmost ball.

(iii) Replace the n assigned to the box with $n + M$.

(iv) Reorder the boxes between adjacent ball groups in ascending order of their assigned numbers.

This discrete-time evolution is mathematically formulated by the ultradiscretization of dhToda-II (udhToda-II) equation:

$$\begin{cases} Q_k^{(n+1)} = \min\left(\sum_{i=1}^{k} Q_i^{(n)} - \sum_{i=1}^{k-1} Q_i^{(n+1)}, E_k^{(n)}\right), \quad k = 1, 2, \ldots, m, \\ Q_k^{(n+1)} + E_k^{(n+M)} = Q_{k+1}^{(n)} + E_k^{(n)}, \quad k = 1, 2, \ldots, m-1, \\ E_0^{(n)} := +\infty, \quad E_m^{(n)} := +\infty. \end{cases} \tag{3.52}$$

Note that, if rules (a)–(d) hold at discrete-time $n = 0$, then they always hold under the discrete-time evolution generated by rules (i)–(iv). Figure 3.2 show an example of discrete-time evolution from $n = 0$ to $n = 6$ for the nBBS-II with $M = 3$.

To demonstrate the soliton behavior in the nBBS-II, we need to clarify the conserved quantities of the udhToda-II equation. Thus, we need to determine the conserved quantities of the dhToda-II equation (3.10) such that their ultradiscretization becomes a conserved quantity in the udhToda-II equation (3.52). The target quantities can be derived by reconsidering the similarity transformation of a Hessenberg matrix generated by the dhToda-II equation (3.10). Although these conserved quantities are equal to the coefficients of the Hessenberg eigenpolynomial, seeking them directly is not easy. Recall that the Hessenberg similarity transformation can be decomposed into multiple bidiagonal LR transformations. Combining this with the Watkins' theorem [35], we can then obtain the target quantities by computing the coefficients of the eigenpolynomial of the sparse band matrix instead of the dense Hessenberg matrix. Details of this process can be found in [15]. Using the resulting conserved quantities, we obtain a theorem concerning soliton behavior in the nBBS-II.

Theorem 3.11 (Yamamoto et al. [36]) *Suppose that m ball groups contain L_1, L_2, $\ldots$, L_m successive balls, which are sufficiently far from each other, at discrete-time $n = 0$. Moreover, let σ be a permutation of $\{1, 2, \ldots, m\}$ such that $L_{\sigma(1)} \le L_{\sigma(2)} \le \cdots \le L_{\sigma(m)}$, and let n^* be the discrete time immediately after the completion of all interactions. Then, the following holds for $n = n^*, n^* + 1, \ldots$:*

$$Q_1^{(n)} = L_{\sigma(1)}, \quad Q_2^{(n)} = L_{\sigma(2)}, \quad \ldots, \quad Q_m^{(n)} = L_{\sigma(m)}.$$

3.6 Concluding Remarks

In our chapter, we first demonstrated that various discrete integrable systems can be related to matrix LR transformations. We also showed that their ultradiscretization of such systems can generate matrix transformations without changing an eigenvalue, and that these transformations can be used to compute the eigenvalue of matrices over min-plus algebra. Finally, we detailed an nBBS with numbered boxes, which arose from the LR perspective of the discrete hungry integrable systems.

Hessenberg matrices including tridiagonal matrices, frequently encountered in this chapter, have a special structure that can be decomposed into products of bidiagonal matrices. This suggests that we can find new discrete integrable systems, and BBSs based on them, by considering other decompositions of Hessenberg matrices from the LR perspective. However, we do not need to only consider Hessenberg matrices in this way. We also do not need to only consider meticulous integrable systems where neighbors are always assumed to affect each other in some way. For example, not all drivers pay continuous attention to their environment. Therefore, tolerant integrable systems may be considered to capture realistic phenomena. We hope that the accumulation of studies on integrable systems and their related LR transformations presented in this chapter improves our ability to represent mobility phenomena.

References

1. F. Baccelli, G. Cohen, G.L. Olsder, J.P. Quadrat, *Synchronization and Linearity* (Wiley, New York, 1992)
2. O.I. Bogoyavlensky, Integrable discretizations of the KdV equation. Phys. Lett. A **134**, 34–38 (1988)
3. A. Common, S. Hafez, Linearization of the relativistic and discrete-time Toda-lattices for particular boundary-conditions. Inverse Probl. **8**, 59–69 (1992)
4. L.A. Dickey, *Soliton Equations and Hamiltonian Systems* (World Scientific, Singapore, 2003)

5. L.D. Faddeev, L.A. Takhtajan, *Hamiltonian Methods in the Theory of Solitons* (Springer, Berlin, 2007)
6. A. Fukuda, E. Ishiwata, M. Iwasaki, Y. Nakamura, On the qd-type discrete hungry Lotka-Volterra system and its application to the matrix eigenvalue algorithm. JSIAM Lett. **1**, 36–39 (2009)
7. A. Fukuda, E. Ishiwata, Y. Yamamoto, M. Iwasaki, Y. Nakamura, Integrable discrete hungry systems and their related matrix eigenvalues. Annal. Mat. Pura Appl. **192**, 423–445 (2013)
8. A. Fukuda, S. Watanabe, A. Hanaoka, M. Iwasaki, Ultradiscrete Lotka-Volterra system computes tropical eigenvalue of symmetric tridiagonal matrices. J. Phys. Conf. Ser. **1218**, 012015 (2019)
9. A. Fukuda, Y. Yamamoto, M. Iwasaki, E. Ishiwata, Y. Nakamura, A Bäcklund transformation between two integrable discrete hungry systems. Phys. Lett. A **375**, 303–308 (2011)
10. A. Fukuda, Y. Yamamoto, M. Iwasaki, E. Ishiwata, Y. Nakamura, On a shifted LR transformation derived from the discrete hungry Toda equation. Monatsh. Math. **170**, 11–26 (2013)
11. A. Fukuda, Y. Yamamoto, M. Iwasaki, E. Ishiwata, Y. Nakamura, Convergence acceleration of the shifted LR transformations for totally nonnegative hessenberg matrices. Appl. Math. **65**, 677–702 (2020)
12. C. Gu (ed.), *Soliton Theory and Its Applications* (Springer, Berlin, 1995)
13. R. Hirota, Conserved quantities of random-time Toda equation. J. Phys. Soc. Jpn. **66**, 283–284 (1997)
14. Y. Itoh, Integrals of a Lotka-Volterra system of odd number of variables. Prog. Theor. Phys. **78**, 507–510 (1987)
15. S. Kakizaki, A. Fukuda, Y. Yamamoto, M. Iwasaki, E. Ishiwata, Y. Nakamura, Conserved quantities of the integrable discrete hungry systems. Discrete Contin. Dyn. Syst. - S **8**, 889–899 (2015)
16. M. Kan, A. Fukuda, S. Watanabe, The ultradiscrete hungry Toda equation and eigenvalues over min-plus algebra. J. Difference Equ. Appl. (2023) https://doi.org/10.1080/10236198.2023.2277714
17. D. Maclagan, B. Sturmfels, *Introduction to Tropical Geometry* (American Mathematical Society, 2015)
18. K. Maruno, K. Kajiwara, M. Oikawa, Casorati determinant solution for the discrete-time relativistic Toda lattice equation. Phys. Lett. A **241**, 335–343 (1998)
19. Y. Minesaki, Y. Nakamura, The discrete relativistic Toda molecule equation and a Padé approximation algorithm. Numer. Algorithms **27**, 219–235 (2001)
20. Y. Nakamura, K. Takasaki, S. Tsujimoto, M. Okado, J. Inokuchi, *Mathematics of Integrable Systems (in Japanese)* (Asakura-Shoten, Tokyo, 2018)
21. Y. Nishiyama, M. Shinjo, K. Kondo, M. Iwasaki, Integrable properties of a variant of the discrete hungry Toda equations and their relationship to eigenpairs of band matrices. East Asia J. Appl. Math. **7**, 785–798 (2018)
22. Y. Ohta, K. Kajiwara, J. Matsukidaira, J. Satsuma, Casorati determinant solution for the relativistic Toda lattice equation. J. Math. Phys. **34**, 5190–5204 (1993)
23. S.N.M. Ruijsenaars, Relativistic Toda systems. Commun. Math. Phys. **133**, 217–247 (1990)
24. H. Rutishauser, *Lectures on Numerical Mathematics* (Birkhäuser, Boston, 1990)
25. M. Sekiguchi, K. Oka, M. Iwasaki, E. Ishiwata, Time-delay version of the integrable discrete Lotka-Volterra system in terms of the LR transformations. IOP SciNotes **2**, 035001 (2021)
26. M. Shinjo, A. Fukuda, K. Kondo, Y. Yamamoto, E. Ishiwata, M. Iwasaki, Y. Nakamura, Discrete hungry integrable systems – 40 years from the Physica D paper by W.W. Symes. Physica D **439**, 133422 (2022)
27. M. Shinjo, M. Iwasaki, K. Kondo, The Kostant-Toda equation and the hungry integrable systems. J. Math. Anal. Appl. **483**, 123627 (2020)
28. M. Shinjo, Y. Nakamura, M. Iwasaki, K. Kondo, Asymptotic analysis of non-autonomous discrete hungry integrable systems. J. Integr. Syst. **3**, xyy001 (2018)
29. Y.B. Suris, A discrete time relativistic Toda lattice. J. Phys. A: Math. Gen. **29**, 451–465 (1996)

30. Y.B. Suris, Integrable discretizations of the Bogoyavlensky lattices. J. Math. Phys. **37**, 3982–3996 (1996)
31. D. Takahashi, J. Satsuma, A soliton cellular automaton. J. Phys. Soc. Jpn. **59**, 3514–3519 (1990)
32. T. Tokihiro, A. Nagai, J. Satsuma, Proof of solitonical nature of box and ball systems by means of inverse ultra-discretization. Inverse Probl. **15**, 1639–1662 (1999)
33. S. Tsujimoto, R. Hirota, S. Oishi, An extension and discretization of Volterra equation I. IEICE Techn. Rep. **92**, 1–3 (1993)
34. S. Watanabe, A. Fukuda, H. Shigitani, M. Iwasaki, Min-plus eigenvalue of tridiagonal matrices in terms of the ultradiscrete Toda equation. J. Phys. A: Math. Theor. **51**, 444001 (2018)
35. D.S. Watkins, Product eigenvalue problems. SIAM Rev. **47**, 3–40 (2005)
36. Y. Yamamoto, N. Minoshita, M. Iwasaki, Discrete relativistic Toda equation from the perspective of shifted LR transformation. Physica D **440**, 133485 (2022)

Part III
Mathematical Methods for Huge Data and Network Analysis

Chapter 4
Numerical Analysis for Data Relationship

Tetsuya Sakurai, Yasunori Futamura, Akira Imakura, and Xiucai Ye

In recent years, a vast amount of data has been accumulated across various fields in industry and academia, and with the rise of artificial intelligence and machine learning technologies, knowledge discovery and high-precision predictions through such data have been demanded. However, real-world data is diverse, including network data that represent relationships, data with multiple modalities or views, data that is distributed across multiple institutions and requires a certain level of information confidentiality. There is also data that requires extracting latent features in complex subspaces for analysis. Therefore, analysis methods that can handle such diversity are needed. In this chapter, we introduce effective methods for such data using novel numerical analysis techniques.

This chapter is organized as follows. Section 4.1 gives an overview of several spectral methods for unsupervised dimensionality reduction and clustering. Section 4.2 describes a recent advanced dimensionality reduction method based on complex moment-based subspace and matrix trace optimization. Section 4.3 shows methods that can utilize data relationships with multiple views simultaneously. In Sect. 4.4, we describe so-called data collaboration analysis that can securely utilize data distributed across multiple institutions.

In this chapter, we denote the numerical dataset of interest by $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]^{\mathrm{T}} \in \mathbb{R}^{n \times m}$, where n is the number of samples (or data objects) and m is the number of features (or attributes). m is also referred to as *dimensionality* of the data objects. We use the symbol $:=$ for definition. We denote $[n] := \{1, 2, \ldots, n\}$. For matrix A, the (i, j)-element is denoted by $[A]_{i,j}$. I_n and $O_{n,m}$ denote the identity matrix of order n and the $n \times m$ zero matrix, respectively.

T. Sakurai (✉) · Y. Futamura · A. Imakura · X. Ye
University of Tsukuba, 1-1-1 Tennodai, Tsukuba, Ibaraki 305-8573, Japan
e-mail: sakurai@cs.tsukuaba.ac.jp

K. Ikeda et al. (eds.), *Advanced Mathematical Science for Mobility Society*,
https://doi.org/10.1007/978-981-99-9772-5_4

4.1 Spectral Methods for Machine Learning

In this section, we describe spectral methods for unsupervised dimensionality reduction and clustering. Spectral methods involve the decomposition or representation of data or relationships in terms of their spectral components. Many spectral methods rely on computing eigenspace of matrices derived from the data in some form. The spectral methods introduced in this section rely on a natively given graph or a graph computed by similarities between the data objects.

Let $\mathcal{G} := (\mathcal{V}, \mathcal{E})$ be a (weighted) undirected graph where $\mathcal{V}$ is the set of vertices and $\mathcal{E}$ is the set of edges. Here we assume that $\mathcal{G}$ is natively given, for example, as in the application in social network analysis. Well-known approaches to form the graph from the data matrix X will be shown at the end of this section. Let W be the adjacency matrix (which is symmetric) of $\mathcal{G}$. $w_{i,j}$ is (i, j) element of W. When $(i, j) \in \mathcal{E}$, a positive weight value is assigned to $w_{i,j}$, otherwise $w_{i,j} = 0$. D is a diagonal matrix whose ith diagonal element is $d_i := \sum_j w_{i,j}$ (for $i \in [n]$). $L := D - W$ is called graph-Laplacian and is known to be positive semi-definite. It is known that the number of zero eigenvalues of L coincides the number of connected components of $\mathcal{G}$. When $\mathcal{G}$ is connected, the values of all elements of the eigenvector corresponding to only zero eigenvalue are constant. In this section, we assume that G is connected, for simplicity. For other cases, refer to [45].

The Laplacian eigenmap [1] is a dimensionality reduction method that minimizes a matrix trace involving L. When one considers dimensionality reduction to ℓ-dimension, the objective function is

$$\min_{U \in \mathbb{R}^{n \times \ell}} \mathrm{Tr}(U^{\mathrm{T}} L U)$$

with the constraint $U^{\mathrm{T}} D U = I_\ell$. The solution of the minimization can be obtained by computing eigenvectors corresponding to smallest ℓ non-zero eigenvalues of the generalized eigenvalue problem $L\boldsymbol{u} = \lambda D\boldsymbol{u}$. This minimization problem can be regarded as a continuous relaxation of the normalize cut problem [45]. The ith row vector of U is regarded as ℓ dimensional representation of the ith data object.

Spectral clustering [34, 45] is one of the most popular clustering methods. This clustering method is a method that applies k-means for low-dimensional representation obtained by the Laplacian eigenmap. The locality preserving projection (LPP) [12] can be regarded as a linear approximation of the Laplacian eigenmap. Its objective function is

$$\min_{Z \in \mathbb{R}^{m \times \ell}} Z^{\mathrm{T}} X^{\mathrm{T}} L X Z$$

with the constraint $Z^{\mathrm{T}} X^{\mathrm{T}} D X Z = I_\ell$. Here we show the algorithm of spectral clustering in Algorithm 1.

The modularity maximization [33] is a popular optimization function for graph clustering and is commonly used for community detection in social network graphs. Modularity maximization is a combinatorial optimization problem. In the modularity

Algorithm 1 Spectral clustering

```
Input: Data matrix $X \in \mathbb{R}^{n\times m}$.
Output: Cluster memberships $C$.
1: Form an undirected graph $G$ from $X$ by e.g. $k$NN.
2: Let $W$ and $D$ be the adjacency matrix and the degree matrix of $G$, respectively. Let $L := D - W$.
3: Compute eigenvectors $\{\boldsymbol{u}\}_{i\in[\ell]}$ corresponding to $\ell$ smallest nonzero eigenvalue $\{\lambda_i\}_{i\in[\ell]}$ of the
   generalized eigenproblem $L\boldsymbol{u} = \lambda D\boldsymbol{u}$.
4: Let $U := [\boldsymbol{u}_1, \boldsymbol{u}_2, \dots, \boldsymbol{u}_\ell]$. Let $\{z_i\}_{i\in[n]}$ be vector of $i$-th row of $X$ such that $U =$
   $[z_1^{\mathrm{T}}; z_2^{\mathrm{T}}; \dots; z_n^{\mathrm{T}}]$.
5: Perform $k-$means clustering to $\{z_i\}_{i\in[n]}$, then determine $C$.
```

maximization, random graph model is introduced. $p_{i,j}$ is the expected weight value of edge between vertex i and j of the random graph model (for unweighted case, $p_{i,j}$ is the probability of the occurrence of an edge). Modularity Q is defined as

$$Q := \frac{1}{\mathrm{Vol}(\mathcal{G})}\sum_{i,j}[w_{i,j} - p_{i,j}]\delta(g_i, g_j).$$

Here, g_i is the cluster index of the ith vertex and $\delta(g_i, g_j) = 1$ where $g_i = g_j$ otherwise 0. Large modularity indicates that the intra-cluster vertices of the graph are more densely connected than a given random graph model. In the Chung–Lu random graph model [33], we define $p_{i,j}$ as $p_{i,j} = \frac{1}{\mathrm{Vol}(\mathcal{G})}D_{i,i}D_{j,j}$ where $\mathrm{Vol}(\mathcal{G}) = \sum_i D_{i,i}$ so that the expected degree of the random graph becomes same as those of $\mathcal{G}$. The modularity matrix is defined as

$$M := W - \frac{1}{\mathrm{Vol}(\mathcal{G})}\boldsymbol{d}\boldsymbol{d}^{\mathrm{T}},$$

where $\boldsymbol{d}$ is the vector formed by the diagonal elements of D. Here we introduce a cluster membership matrix U. $[U]_{i,j} = 1$ when the ith vertex belongs to the jth cluster and $[U]_{i,j} = 0$ otherwise. Using this, modularity can be written as

$$Q = \mathrm{Tr}(U^{\mathrm{T}}MU).$$

Here we omitted the constant factor. When we relax the binary constraint and allow the continuous values (with constraint $U^{\mathrm{T}}U = I_n$), the solution is obtained by computing the eigenvectors corresponding to the ℓ largest eigenvalues of the standard eigenvalue problem

$$M\boldsymbol{u} = \lambda\boldsymbol{u}.$$

This dimensionality reduction method is not as widely used as the Laplacian eigenmap, but could be an alternative to it. It is proposed as an anomaly detection method for densely connected subgraphs [9].

Now we introduce methods for constructing a graph from the data matrix X. One of the most popular approaches is to make the kNN graph. Given pairwise similarity, kNN graph we set $(i, j) \in \mathcal{E}$ if vertex i is a k-nearest neighbor vertex of j. Because the edge occurrence is not necessarily symmetric, some process for symmetrization is performed [45]. The complexity of a brute force approach to compute pairwise similarity (or distance) is $O(n^2)$ that is not tractable for large n. There are approximation methods such as recursive Lanczos bisection [2] and kNN descent [5] that are aimed at reducing computational complexity.

4.2 Complex Moment-Based Supervised Eigenmap for Dimensionality Reduction

Increasing the number of features (dimensionality) seems to lead to better performance. However, in practice, adding more features leads to worse performance, i.e. the curse of dimensionality. Dimensionality reduction reduces the number of features (dimensionality) to avoid the curse of dimensionality. Using dimensionality reduction methods, we can speed up algorithms, reduce the risk of overfitting and improve the accuracy of prediction results.

In this section, we briefly introduce dimensionality reduction methods based on matrix trace optimization. These methods use a few eigenvectors to construct a low-dimensional space while preserving certain properties of the original data. We also introduce a complex moment-based supervised eigenmap [17] that uses a large number of eigenvectors to improve recognition performance.

4.2.1 Dimensionality Reduction Based on a Matrix Trace Optimization

Let m and n be the number of the features and samples for training dataset $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]^{\mathrm{T}} \in \mathbb{R}^{n \times m}$. Here, we consider linear and nonlinear dimensionality reduction methods that construct low-dimensional data $Y = [\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n]^{\mathrm{T}} \in \mathbb{R}^{n \times \ell}$, which retain some of the properties of the original data. Specifically, linear dimensionality reduction methods reduce the original data X to the low-dimensional data Y using a linear map $B \in \mathbb{R}^{m \times \ell}$, i.e.

$$Y = XB.$$

In each dimensionality reduction method based on a matrix trace optimization, a symmetric matrix $A_1 \in \mathbb{R}^{m \times m}$ and a symmetric positive definite matrix $A_2 \in \mathbb{R}^{m \times m}$ are defined, respectively. Then, the linear map B is formulated by the minimization or maximization of a matrix trace:

$$\min_{B \in \mathbb{R}^{m \times \ell}} \operatorname{Tr}\left(B^{\mathrm{T}} A_1 B\right) \quad \text{or} \quad \max_{B \in \mathbb{R}^{m \times \ell}} \operatorname{Tr}\left(B^{\mathrm{T}} A_1 B\right) \quad \text{s.t.} \quad B^{\mathrm{T}} A_2 B = I_\ell.$$

This is solved as ℓ eigenvectors of the corresponding generalized eigenvalue problem:

$$A_1 \boldsymbol{t}_i = \lambda_i A_2 \boldsymbol{t}_i. \tag{4.1}$$

Here, we have $B = [\boldsymbol{t}_1, \boldsymbol{t}_2, \dots, \boldsymbol{t}_\ell]$.

Principal component analysis (PCA) [24, 39] and locality preserving projections (LPP) [12] are two of the typical unsupervised dimensionality reduction methods in this class. PCA aims to maximize the variance of the projected vectors, while LPP aims to preserve the local similarity of the original data. Discriminant analysis is the typical supervised method that maximizes the between-class scatter and reduces the within-class scatter. A family of discriminant analysis methods are proposed for supervised dimensionality reduction, including Fisher discriminant analysis (FDA) [7, 8], local FDA (LFDA) [43], semi-supervised LFDA (SELF) [44] and locality adaptive discriminant analysis (LADA) [28].

Nonlinear dimensionality reduction methods, which use a nonlinear map and the kernel trick [42], are widely used as improvements over linear dimensionality reduction methods. Nonlinear dimensionality reduction methods transform the original data X to $\phi(X) = [\phi(\boldsymbol{x}_1), \phi(\boldsymbol{x}_2), \dots, \phi(\boldsymbol{x}_n)]^{\mathrm{T}}$ with a nonlinear kernel function and reduce the dimension of $\phi(X)$ using a nonlinear map $\widetilde{B}$ such that $Y = \phi(X)\widetilde{B}$. With appropriate nonlinear functions, nonlinear dimensionality reduction methods are expected to improve the recognition performance.

In general, we set $\widetilde{B} = \phi(X)^{\mathrm{T}} \widehat{B}$ with $\widehat{B} \in \mathbb{R}^{n \times \ell}$ and directly set the Gram matrix $K = \phi(X)\phi(X)^{\mathrm{T}} \in \mathbb{R}^{n \times n}$ without computing $\phi(X)$ to reduce the computational costs. The Gaussian kernel, polynomial kernel and sigmoid kernel are commonly used as the kernel functions.

4.2.2 A Complex Moment-Based Supervised Eigenmap

As written in Sect. 4.2.1, most of the existing dimensionality reduction methods use only ℓ eigenvectors to construct the low-dimensional space with a dimension ℓ, which may lead to loss of useful information for achieving successful classification. To overcome the deficiency of information loss, recently, a complex moment-based supervised eigenmap (CMSE) for dimensionality reduction has been proposed [17]. CMSE allows us to achieve better recognition performance by using a complex moment-based subspace that includes $d > \ell$ eigenvectors, where d can be set independently of the dimension ℓ of the low-dimensional space.

The basic concepts of CMSE for high recognition performance are summarized as follows:

- Use the complex moment-based subspace $\mathcal{S}_\Omega$, which is equivalent to the invariant subspace with respect to the multiple eigenvectors.
- Use the novel minimization problem that combines the matrix trace derived from the dimensionality reduction methods and the squared error straightforwardly using the ground truth data $Z \in \mathbb{R}^{n \times \ell}$.

Let A_1 and A_2 be the matrices used in a given dimensionality reduction method such as LPP or LFDA. Based on the concept of complex moment-based eigensolvers [14, 40, 41], we also let $\mathcal{S}_\Omega$ be a complex moment-based subspace with respect to a given real interval $\Omega = [a, b] \subset \mathbb{R}$ defined by

$$\mathcal{S}_\Omega = \mathcal{R}(S), \quad S = [S_0, S_1, \ldots, S_{M-1}],$$
$$S_k := \frac{1}{2\pi \mathrm{i}} \oint_\Gamma z^k (zA_2 - A_1)^{-1} A_2 V \mathrm{d}z, \tag{4.2}$$

where $L, M \in \mathbb{N}_+$, $V \in \mathbb{R}^{m \times L}$ and Γ is a positively oriented Jordan curve around Ω. Then, the subspace $\mathcal{S}_\Omega = \mathcal{R}(S)$ is equivalent to the subspace spanned by eigenvectors $\boldsymbol{t}_i$ corresponding to eigenvalues $\lambda_i \in \Omega$ of (4.1), that is,

$$\mathcal{S}_\Omega = \mathcal{R}(S) = \{\boldsymbol{t}_i | \lambda_i \in \Omega\}.$$

Then, using $d = \mathrm{rank}(\mathcal{S}_\Omega) \geq \ell$ eigenvectors, CMSE introduces the following minimization problem on $\mathcal{S}_\Omega$ to obtain the linear map $B \in \mathbb{R}^{m \times \ell}$:

$$\min_{B=[\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_\ell], \boldsymbol{b}_i \in \mathcal{S}_\Omega} E(B) \quad \text{s.t.} \quad B^{\mathrm{T}} A_2 B = I_\ell, \tag{4.3}$$

with

$$E(B) = (1 - \mu)\mathrm{Tr}\left(B^{\mathrm{T}} f(A_1) B\right) + \mu \|Z - XB\|_{\mathrm{F}}^2,$$

where the objective function $E(B)$ combines a matrix trace derived from dimensionality reduction methods and a squared error straightforwardly using the ground truth data Z. Here, $\mu \in [0, 1]$ is a weight parameter for both terms and $f(\cdot)$ is a (meromorphic) weight function of each eigenvector for minimization.

In (4.3), the column vectors of the linear map B are constrained by A_2-orthonormal bases of the complex moment-based subspace $\mathcal{S}_\Omega$. Let $U \in \mathbb{R}^{m \times d}$ be an A_2-orthogonal matrix ($U^{\mathrm{T}} A_2 U = I_d$) whose columns are A_2-orthonormal bases of the complex moment-based subspace $\mathcal{S}_\Omega$ and $T = U^{\mathrm{T}} A_1 U$. Then, the minimization problem (4.3) can be written as

$$\min_{C \in \mathbb{R}^{d \times \ell}} \left\| \begin{bmatrix} \mu^{1/2} Z \\ O_{d,\ell} \end{bmatrix} - \begin{bmatrix} \mu^{1/2} X U \\ (1-\mu)^{1/2} f(T)^{1/2} \end{bmatrix} C \right\|_{\mathrm{F}}^2 \quad \text{s.t.} \quad C^{\mathrm{T}} C = I_\ell, \tag{4.4}$$

where $B = UC$. A minimization problem with an orthogonal constraint (4.4) is called an unbalanced orthogonal Procrustes (UOP) problem, which is solved using an iterative method [6, 38].

```
Algorithm 2 Complex moment-based supervised eigenmap (CMSE)
Input: Training dataset: $X \in \mathbb{R}^{n\times m}$, $Z \in \mathbb{R}^{n\times \ell}$ and parameters: $L, M, N \in \mathbb{N}_+, \delta \in \mathbb{R}, V \in \mathbb{R}^{m\times L}$, $(z_j, \omega_j)$ for $j = 1, 2, \ldots, N/2$, $\Omega = [a, b]$, $\mu$, $f(\cdot)$.
Output: Linear eigenmap $B \in \mathbb{R}^{m\times \ell}$
1: Construct a matrix pencil $(A_1, A_2)$ from the training dataset $X$ (and $Z$ if required)
2: Compute $\widehat{S}_k = \sum_{j=1}^{N/2} \mathrm{Re}(\omega_j z_j^k (z_j A_2 - A_1)^{-1} A_2 V)$, and set $\widehat{S} = [\widehat{S}_0, \widehat{S}_1, \ldots, \widehat{S}_{M-1}]$
3: Compute a low-rank approximation of $\widehat{S}$ using the threshold $\delta$:
   $\widehat{S} = [\widehat{U}, \widehat{U}'][\widehat{\Sigma}, O; O, \widehat{\Sigma}'][\widehat{W}, \widehat{W}'^{\mathrm{T}}] \approx \widehat{U}\widehat{\Sigma}\widehat{W}^{\mathrm{T}}$ such that $\widehat{U}^{\mathrm{T}} A_2 \widehat{U} = I_{\widehat{d}}$
4: Solve UOP problem (4.6) and set $B = \widehat{U}\widehat{C}$
```

In practice, the contour integral (4.2) is approximated by a numerical integration rule such as the N-point trapezoidal rule, as follows:

$$\widehat{S}_k = 2\sum_{j=1}^{N/2} \mathrm{Re}(\omega_j z_j^k (z_j A_2 - A_1)^{-1} A_2 V), \tag{4.5}$$

using symmetric property. In addition, to improve the numerical stability, we apply a low-rank approximation of $\widehat{S}$ with a truncated singular value decomposition on an A_2-inner product:

$$\widehat{S} = [\widehat{U}, \widehat{U}'] \begin{bmatrix} \widehat{\Sigma} & \\ & \widehat{\Sigma}' \end{bmatrix} \begin{bmatrix} \widehat{W}^{\mathrm{T}} \\ \widehat{W}'^{\mathrm{T}} \end{bmatrix} \approx \widehat{U}\widehat{\Sigma}\widehat{W}^{\mathrm{T}}, \quad \widehat{U}^{\mathrm{T}} A_2 \widehat{U} = I_{\widehat{d}}, \quad \widehat{W}^{\mathrm{T}}\widehat{W} = I_{\widehat{d}},$$

where $\widehat{\Sigma}$ is a diagonal matrix whose diagonal entries are the larger part of the singular values, i.e., $\sigma_i/\sigma_1 \geq \delta$ $(\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{LM})$. The columns of $\widehat{U}$ and $\widehat{W}$ are the corresponding left and right singular vectors. Let $\widehat{d}$ be a numerical rank, $\sigma_{\widehat{d}}/\sigma_1 \geq \delta > \sigma_{\widehat{d}+1}/\sigma_1$. Then, the UOP problem (4.4) is rewritten as

$$\min_{\widehat{C}\in\mathbb{R}^{\widehat{d}\times\ell}} \left\| \begin{bmatrix} \mu^{1/2} Z \\ O_{\widehat{d},\ell} \end{bmatrix} - \begin{bmatrix} \mu^{1/2} X\widehat{U} \\ (1-\mu)^{1/2} f(\widehat{T})^{1/2} \end{bmatrix} \widehat{C} \right\|_{\mathrm{F}}^2 \quad \text{s.t.} \quad \widehat{C}^{\mathrm{T}}\widehat{C} = I_\ell, \tag{4.6}$$

where $\widehat{T} = \widehat{U}^{\mathrm{T}} A_1 \widehat{U}$ and the map is obtained from $B = \widehat{U}\widehat{C}$.

The algorithm of CMSE is summarized in Algorithm 2. One of the most time-consuming parts of CMSE is computing the solutions of the $N/2$ linear systems with L right-hand sides in (4.5) and Step 2 of Algorithm 2 as follows:

$$(z_j A_2 - A_1) P_j = A_2 V, \quad j = 1, 2, \ldots, N/2.$$

Since CMSE has hierarchical parallelism for solving these linear systems, CMSE shows a high parallel efficiency demonstrated in [48].

4.3 Multi-view Data Analysis

Multi-view data are ubiquitous in real-world applications. Multi-view data refers to data that includes multiple sets of features or representations, each providing a different perspective or view of the objects or entities being observed. For instance, pictures often have textual tags and descriptions and the same semantic meaning can be represented in multilingual forms. Each individual view has its specific property in these data, while different views often contain complementary information [47]. Multi-view learning can be roughly divided into supervised and unsupervised methods. Here, we mainly introduce the unsupervised method, i.e., multi-view clustering, which has emerged as a powerful tool for exploring the underlying structure of data from different sources.

Next, we introduce three kinds of multi-view clustering methods, including multi-kernel learning, multi-view graph clustering and multi-view subspace clustering.

4.3.1 Multi-Kernel Learning

Multi-kernel learning has been widely applied in order to deal with multi-view data, which intends to optimally combine a group of predefined kernels in order to improve clustering performance [47]. The general method for multi-view data is to use a linear combination of several kernel functions. Since the weights of different views are important, the weights of different kernels should be taken into consideration. An auto-weighted multi-kernel method has been proposed to weigh the views and kernels simultaneously [51]. First, Kernel Principal Component Analysis (KPCA) is used on each view to reduce the dimension of multi-view data. Then, the designed weighted Gaussian kernel is applied to the low-dimensional multi-view data. The weighted Gaussian kernel integrates the advantage of the Gaussian kernel and the Polynomial kernel, which is formulated as

$$K(x, y) = \left[\exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right) + R\right]^d, \quad \forall R \geq 0, d \geq 0, d \in N.$$

The above formula can be expanded based on the binomial theorem as follows.

$$\begin{aligned} K(x, y) &= \sum_{s=0}^{d} \binom{d}{s} R^{d-s} \exp\left(-\frac{s\|x-y\|^2}{2\sigma^2}\right) \\ &= R^d + \sum_{s=1}^{d} \binom{d}{s} R^{d-s} \exp\left(-\frac{\|x-y\|^2}{2\sigma^2/s}\right). \end{aligned}$$

Thus, the weighted Gaussian kernel is a combination of d Gaussian kernels with different widths in the range from σ^2 to σ^2/s. Weight R^{d-s} is used to reflect the importance of the $d - s$ kernel and enlarge the linear translation of distance between points.

For multi-view data with m views, n samples and k clusters, the objective function based on the K-means and weighted Gaussian kernel is formulated as

$$\min \sum_{v=1}^{m} \sum_{i=1}^{n} \omega_{iv} \sum_{j=1}^{k} \delta_{ij} \|\phi^v(x_i^v) - \phi^v(c_j^v)\|^2,$$
$$s.t., \omega_{iv} \geq 0, \prod_{v} \omega_{iv} = 1,$$

where c_j^v is the cluster center, and δ_{ij} is the indicator variable with $\delta_{ij} = 1$ if $x_i \in c_j$, otherwise $\delta_{ij} = 0$. This step drives the weight of each view and cluster center. After finite iterations, it arrives at the final clustering result.

By replacing the weighted Gaussian kernel $K(x, y) = \phi(x) \cdot \phi(y)$, the above formula can be rewritten as

$$\min \sum_{v=1}^{m} \sum_{i=1}^{n} \omega_{iv} \sum_{j=1}^{k} 2\delta_{ij}[(1 + R)^d - K(x_i^v, c_j^v)],$$
$$s.t., \omega_{iv} \geq 0, \prod_{v} \omega_{iv} = 1.$$

The above formula inherits the properties of K-means and kernel, while the weighted Gaussian kernel integrates the advantages of the Gaussian kernel and Polynomial kernel.

4.3.2 Multi-view Graph Clustering

The objective of multi-view graph clustering is to find a fusion graph across all views and then use other methods such as spectral clustering on the fusion graph to obtain the final clustering result. Here, we introduce a parameter-free Auto-weighted Multiple Graph Learning (AMGL) method [37], which implements the automatic allocation of weights by modifying the traditional spectral clustering model without any hyperparameters.

In spectral clustering, based on the Laplacian matrix, the objective function can be defined as follows.

$$\min_{F^{\mathrm{T}}F=I} \mathrm{Tr}(F^{\mathrm{T}} L F).$$

Based on the above formula and replacing L with L^v, i.e., the Laplacian matrix in each view v, a new general framework for multiple graph learning is proposed.

$$\min_{F \in C} \sum_{v=1}^{m} \sqrt{\mathrm{Tr}(F^{\mathrm{T}} L^{v} F)},$$

where no weight factors are explicitly defined. By taking the derivative of the Lagrange function of the above formula, the weight of each view α^v can be obtained based on F as

$$\alpha^{v} = \frac{1}{2\sqrt{\mathrm{Tr}(F^{\mathrm{T}} L^{v} F)}}. \tag{4.7}$$

Then, the objective function of the AMGL method is set as

$$\min_{F \in C} \sum_{v=1}^{m} \alpha^{v} \mathrm{Tr}(F^{\mathrm{T}} L^{v} F).$$

In the above formula, F will be continuously used to update α^v according to Eq. (4.7), which inspires taking an alternating optimization strategy to compute F and α^v iteratively.

4.3.3 *Multi-view Subspace Clustering*

Multi-view subspace clustering is to learn a unified representation or a latent space from all view data. Then, the unified representation is fed into an off-the-shelf clustering model to obtain the clustering results. Here, we introduce Non-Negative Matrix Factorization (NMF)-based method.

NMF aims to find two non-negative matrices $U \in \mathbb{R}^{n\times p}$ and $V \in \mathbb{R}^{p\times m}$ to adequately approximate the original matrix $X \in \mathbb{R}^{n\times m}$. The reconstruction processes can be formulated as the following optimization problem.

$$\min_{U,V} \|X - UV\|_F^2, \quad s.t., U \geq 0, V \geq 0.$$

Here, U is termed as the basis matrix, while V is the indicator matrix, and p denotes the desired reduced dimension. Due to the non-negative constraints, it can learn a part-based representation.

For multi-view data, the objective is to combine multi-view information in the NMF framework. Generally, a common indicator matrix V^* is enforced in the NMF among different views to perform multi-view clustering. One of the widely used methods is to push each view-dependent indicator matrix V^v toward a common indicator matrix V^* [29]. The optimization problem is formulated as

$$\min_{U^v, V^v, v=1,\ldots,m} \sum_{v=1}^{m} \|X^v - U^v V^v\|_F^2 + \lambda_v \|V^v - V^*\|_F^2,$$
$$s.t., \forall 1 \geq k \geq K, \|U^v_{\cdot,k}\|_1 = 1, U^v, V^v, V^* \geq 0.$$

The constraint $\|U^v_{\cdot,k}\|_1 = 1$ is to guarantee that V^v is within the same range for different views. After obtaining the consensus matrix V^*, the cluster label of the data point i can be computed as $\arg\max_k V^*_{i,k}$.

4.4 Data Collaboration Analysis

In various real-world applications, there is a growing demand for the integrated data analysis, in which the datasets are owned by multiple parties in a distributed manner, and they are collaboratively analyzed. For example, in medical data analysis for rare diseases, it was reported that when the analysis is conducted using only data from a single institution, the accuracy is insufficient because of the small sample size [30]. However, sharing the original medical data is difficult because of privacy concerns, and even if it could be achieved, we have to pay huge costs for cross-institutional cross-border communications. Therefore, methods to achieve privacy-preserving analysis in which datasets are collaboratively analyzed without sharing the original data are attracting attention.

In this section, we briefly introduce typical privacy-preserving methods and describe a data collaboration analysis.

4.4.1 *Privacy-Preserving Integrated Data Analysis*

A compelling application scenario for privacy-preserving integrated data analysis is medical data analysis involving multiple institutions. Consider a scenario where there are several municipalities, each with its own set of patients or data samples. Within each municipality, patients receive medical examinations or treatments at multiple medical institutions. There are a variety of medical data (i.e., features), such as red blood cell count and white blood cell count, assigned to each patient and the type of data varies by institution. Therefore, medical data are partitioned by samples into multiple municipalities (i.e., horizontal data partitioning) and by features into multiple medical institutions (i.e., vertical data partitioning).

If the analysis is conducted using only data in a single institution (*local analysis*), the accuracy may be insufficient because of the small sample size and limited features. By integrating data from multiple medical institutions in multiple municipalities (*centralized analysis*), one can achieve a highly accurate analysis; however, data sharing is difficult because of the perspective of data confidentiality. Thus, privacy-preserving integrated data analysis for horizontally and vertically partitioned data is essential.

Cryptographic computation is one of the most well-known methods used for ensuring privacy preservation [4, 11, 23]. Cryptographic methods can compute a function over distributed data while retaining the privacy of the data. Any given function can be computed by applying fully homomorphic encryption [10]. However, this method is not feasible for a machine learning model construction of large datasets because of the large computational cost even with the latest implementations [3, 50].

In the context of a model construction, a typical technology of a privacy-preserving integrated data analysis is federated learning systems introduced by Google. The concept of federated learning systems was first proposed by Google [25] typically for Android phone model updates [31]. Federated learning is primarily based on (deep) neural network and updates the model iteratively.

To update the model, federated stochastic gradient descent (FedSGD) and federated averaging (FedAvg) are typical strategies [31]. FedSGD is a direct extension of the stochastic gradient descent method. In each iteration of the gradient descent method, each party locally computes a gradient from the shared model using the local dataset and sends it to the server. The shared gradients are averaged and used to update the model. Instead, in FedAvg, each party performs more than one batch update using the local dataset, and sends the updated model to the server. Then, the shared models are averaged to update. Federated learning including more recent methods, such as FedProx [27] and FedCodl [36], require cross-institutional communication in each iteration. For more details, we refer to [26, 46] and references therein.

Recently, non-model share-type federated learning called *data collaboration (DC) analysis* has been proposed [18, 21]. Instead of the above model share-type federated learning, the DC analysis centralizes the dimensionality-reduced *intermediate representation*. The centralized intermediate representations are transformed to incorporable forms called *collaboration representations*. Then, the collaborative representation is analyzed as a single dataset. The DC analysis does not require iterative cross-institutional communications.

The DC analysis has been developed to interpretable model construction [15], novelty detection [22], feature selection [49] and survival analysis [20]. Privacy and accuracy of DC analysis were analyzed in [13]. In addition, identifiability, which is essential for analyzing personal information, of the shared intermediate representation was analyzed [19]. The paper [19] proposed a non-readily identifiable DC analysis that realizes privacy-preserving analysis sharing only non-readily identifiable intermediate representations.

4.4.2 Data Collaboration Analysis

Let m and n denote the numbers of features (dimensionality of each data) and training data samples. Let $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]^{\mathrm{T}} \in \mathbb{R}^{n \times m}$ and $Y = [\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots,$

$\boldsymbol{y}_n]^{\mathrm{T}} \in \mathbb{R}^{n\times\ell}$ be the training dataset and the corresponding ground truth or label. Here, for privacy-preserving integrated data analysis on multiple parties, we introduce the algorithm of the DC analysis for supervised learning of horizontal partitioned data, that is, data samples are partitioned into c parties as follows:

$$X = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_c \end{bmatrix}, \quad Y = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_c \end{bmatrix}.$$

Then, the ith party has a partial dataset and the corresponding ground truth,

$$X_i \in \mathbb{R}^{n_i\times m}, \quad Y_i \in \mathbb{R}^{n_i\times\ell},$$

where $n = \sum_{i=1}^{c} n_i$. Note that DC analysis is applicable to the datasets with partially common features [32] and horizontal and vertical partitioned data [21].

DC analysis operates in two roles: *worker* and *master*. Workers have the private dataset X_i and corresponding ground truth Y_i, which must be analyzed without sharing X_i. Master supports the collaboration analysis.

First, all workers generate the same anchor data $X^{\mathrm{anc}} \in \mathbb{R}^{r\times m}$, which is shareable data consisting of public data or dummy data that are randomly constructed. Although, random anchor data functions well for DC analysis in general [18, 21, 22], using an anchor data whose distribution is close to that of the raw dataset can improve the recognition performance [16]. Then, each worker constructs intermediate representations,

$$\widetilde{X}_i = f_i(X_i) \in \mathbb{R}^{n_i\times\widetilde{m}_i}, \quad \widetilde{X}_i^{\mathrm{anc}} = f_i(X^{\mathrm{anc}}) \in \mathbb{R}^{r\times\widetilde{m}_i},$$

where $\widetilde{m}_i < m$, and centralizes them to the master. Here, we can use non-supervised and supervised dimensionality reduction methods, e.g., in Sect. 4.2.

At the master side, mapping function g_i for the collaboration representation is constructed satisfying $g_i(\widetilde{X}_i^{\mathrm{anc}}) \approx g_{i'}(\widetilde{X}_{i'}^{\mathrm{anc}})$ $(i, i' = 1, 2, \ldots, c)$ in some sense. In practice, g_i is set as a linear function $g_i(\widetilde{X}_i^{\mathrm{anc}}) = \widetilde{X}_j^{\mathrm{anc}} G_j$ with $G_i \in \mathbb{R}^{\widetilde{m}_i\times\widehat{m}}$ and is constructed using the following minimal perturbation problem:

$$\min_{E_i, G_i' (i=1,2,\ldots,c), \|Z\|_{\mathrm{F}}=1} \sum_{i=1}^{c} \|E_i\|_{\mathrm{F}}^2 \quad \text{s.t. } (\widetilde{X}_i^{\mathrm{anc}} + E_i)G_i' = Z.$$

This can be solved by a singular value decomposition (SVD)-based algorithm for total least squares problems. Let

$$[\widetilde{X}_1^{\mathrm{anc}}, \widetilde{X}_2^{\mathrm{anc}}, \ldots, \widetilde{X}_c^{\mathrm{anc}}] \approx U_{\widehat{m}} \Sigma_{\widehat{m}} V_{\widehat{m}}^{\mathrm{T}} \tag{4.8}$$

Algorithm 3 Data collaboration (DC) analysis

```
Input: $X_i \in \mathbb{R}^{n_i \times m}$, $Y_i \in \mathbb{R}^{n_i \times \ell}$, and $X_i^{\mathrm{test}}$ individually
Output: $Y_i^{\mathrm{pred}}$ $(i = 1, 2, \dots, c)$.

        Worker-side $(i = 1, 2, \dots, c)$
 1: Generate $X^{\mathrm{anc}}$ and share to all workers
 2: Generate $f_{i,j}$
 3: Compute $\widetilde{X}_i = f_i(X_i)$ and $\widetilde{X}_i^{\mathrm{anc}} = f_i(X^{\mathrm{anc}})$
 4: Share $\widetilde{X}_i$, $\widetilde{X}_i^{\mathrm{anc}}$, and $Y_i$ to master

                              Master-side
 5:                   $\searrow$ Obtain $\widetilde{X}_i$, $\widetilde{X}_i^{\mathrm{anc}}$, and $Y_i$ for all $i$
 6:                     Compute $G_i$ from $\widetilde{X}_i^{\mathrm{anc}}$ for all $i$ by (4.8) and (4.9)
 7:                     Compute $\widehat{X}_i = \widetilde{X}_i G_i$ for all $i$, and set $\widehat{X}$
 8:                     Analyze $\widehat{X}$ to obtain $h$ such that $Y \approx h(\widehat{X})$
 9:                   $\swarrow$ Return $G_i$ and $h$ to each worker

        Worker-side $(i = 1, 2, \dots, c)$
10: Obtain $G_i$ and $h$
11: Compute $Y_i^{\mathrm{pred}} = h(f_i(X_i^{\mathrm{test}})G_i)$
```

be the rank $\widehat{m}$ approximation based on SVD. Then, the target matrix G_i is obtained as follows:

$$G_i = (\widetilde{X}_i^{\mathrm{anc}})^{\dagger} U_{\widehat{m}} C, \tag{4.9}$$

where $\dagger$ denotes the Moore–Penrose inverse and $C \in \mathbb{R}^{\widehat{m} \times \widehat{m}}$ is a nonsingular matrix, for example, $C = I_{\widehat{m}}$ and $C = \Sigma_{\widehat{m}}$ are used in practice. The collaboration representations are analyzed as a single dataset, that is,

$$Y \approx h(\widehat{X}), \quad \widehat{X} = [\widehat{\boldsymbol{x}}_1, \widehat{\boldsymbol{x}}_2, \dots, \widehat{\boldsymbol{x}}_n]^{\mathrm{T}} = \begin{bmatrix} \widehat{X}_1 \\ \widehat{X}_2 \\ \vdots \\ \widehat{X}_c \end{bmatrix} = \begin{bmatrix} \widetilde{X}_1 G_1 \\ \widetilde{X}_2 G_2 \\ \vdots \\ \widetilde{X}_c G_c \end{bmatrix} \in \mathbb{R}^{n \times \widehat{m}}$$

with the shared ground truth Y_i using some supervised machine learning or the deep learning methods for constructing the model function h of the collaboration representation $\widehat{X}$. Functions g_i and h are returned to the ith worker.

Let $X_i^{\mathrm{test}} \in \mathbb{R}^{s_i \times m}$ be a test dataset of the ith party. For the prediction phase, the prediction result Y_i^{pred} of X_i^{test} is obtained by the following equation:

$$Y_i^{\mathrm{pred}} = h(g_i(f_i(X_i^{\mathrm{test}})))$$

through the intermediate and collaboration representations.

The algorithm of the DC analysis is summarized in Algorithm 3, where g_i is set by (4.9). The DC analysis requires only three cross-institutional communications, Steps 1, 4 and 9 in Algorithm 3. This is a major advantage over federated learning.

As in [13], the DC analysis has the following double privacy layer for the protection of private data X_i:

- No one can infer the private data X_i under the protocol;
- Even if f_i is stolen, the private data X_i is still protected regarding ε-DR privacy [35].

Under the protocol (Algorithm 3), the function f_i for the intermediate representation is private and cannot be inferred by others because both the input and output of f_i are not possessed. Therefore, it is impossible to infer the original data X_i only from the shared intermediate representation $\widetilde{X}_i = f_i(X_i)$. In addition, the function f_i is set to a dimensionality reduction function such that $\widetilde{m}_i < m$. Therefore, it is impossible to obtain the original data X_i from $\widetilde{X}_i = f_i(X_i)$ even when using f_i.

References

1. M. Belkin, P. Niyogi, Laplacian eigenmaps and spectral techniques for embedding and clustering. Adv. Neural Inf. Proc. Syst. **14**, 585–591 (2001)
2. J. Chen, H.-R. Fang, Y. Saad, Fast approximate KNN graph construction for high dimensional data via recursive Lanczos bisection. J. Mach. Learn. Res. **10**, 1989–2012 (2009)
3. I. Chillotti, N. Gama, M. Georgieva, M. Izabachene, Faster fully homomorphic encryption: bootstrapping in less than 0.1 seconds, in *International Conference on the Theory and Application of Cryptology and Information Security* (Springer, 2016), pp. 3–33
4. H. Cho, D.J. Wu, B. Berger, Secure genome-wide association analysis using multiparty computation. Nat. Biotechnol. **36**(6), 547–551 (2018)
5. W. Dong, C. Moses, K. Li, Efficient k-nearest neighbor graph construction for generic similarity measures, in *Proceedings of the 20th International Conference on World Wide Web* (2011), pp. 577–586. https://doi.org/10.1145/1963405.1963487
6. L. Eldén, H. Park, A procrustes problem on the Stiefel manifold. Numer. Math. **82**(4), 599–619 (1999)
7. R.A. Fisher, The use of multiple measurements in taxonomic problems. Ann. Hum. Genet. **7**(2), 179–188 (1936)
8. K. Fukunaga, *Introduction to Statistical Pattern Recognition* (Academic, 2013)
9. Y. Futamura, X. Ye, A. Imakura, T. Sakurai, Spectral anomaly detection in large graphs using a complex moment-based eigenvalue solver. ASCE-ASME J. Risk Uncertain. Eng. Syst. A **6**(2) (2020). https://doi.org/10.1061/ajrua6.0001054
10. C. Gentry, Fully homomorphic encryption using ideal lattices, in *Proceedings of the 41st annual ACM Symposium on Theory of Computing* (2009), pp. 169–178
11. R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, J. Wernsing, Cryptonets: applying neural networks to encrypted data with high throughput and accuracy, in *Proceedings of the 33rd International Conference on Machine Learning* (2016), pp. 201–210
12. X. He, P. Niyogi, Locality preserving projections. Adv. Neural Inf. Proc. Syst. **16**, 153–160 (2004)
13. A. Imakura, A. Bogdanova, T. Yamazoe, K. Omote, T. Sakurai, Accuracy and privacy evaluations of collaborative data analysis, in *Proceedings of the 2nd AAAI Workshop on Privacy-Preserving Artificial Intelligence* (2021)

14. A. Imakura, L. Du, T. Sakurai, Relationships among contour integral-based methods for solving generalized eigenvalue problems. Jpn. J. Ind. Appl. Math. **33**(3), 721–750 (2016)
15. A. Imakura, H. Inaba, Y. Okada, T. Sakurai, Interpretable collaborative data analysis on distributed data. Expert Syst. Appl. **177**, 114891 (2021)
16. A. Imakura, M. Kihira, Y. Okada, T. Sakurai, Another use of SMOTE for interpretable data collaboration analysis. Expert Syst. Appl. **228**, 120385 (2023)
17. A. Imakura, M. Matsuda, X. Ye, T. Sakurai, Complex moment-based supervised eigenmap for dimensionality reduction, in *Proceedings of the 33rd AAAI Conference on Artificial Intelligence* (2019), pp. 3910–3918
18. A. Imakura, T. Sakurai, Data collaboration analysis framework using centralization of individual intermediate representations for distributed data sets. ASCE-ASME J. Risk Uncertain. Eng. Syst. A **6**, 04020018 (2020)
19. A. Imakura, T. Sakurai, Y. Okada, T. Fujii, T. Sakamoto, H. Abe, Non-readily identifiable data collaboration analysis for multiple datasets including personal information. Inf. Fusion **98**, 101826 (2023)
20. A. Imakura, R. Tsunoda, R. Kagawa, K. Yamagata, T. Sakurai, DC-COX: data collaboration Cox proportional hazards model for privacy-preserving survival analysis on multiple parties. J. Biomed. Inf. **137**, 104264 (2023)
21. A. Imakura, X. Ye, T. Sakurai, Collaborative data analysis: non-model sharing-type machine learning for distributed data, in *Knowledge Management and Acquisition for Intelligent Systems* (2021), pp. 14–29
22. A. Imakura, X. Ye, T. Sakurai, Collaborative novelty detection for distributed data by a probabilistic method, in *Asian Conference on Machine Learning* (2021), pp. 932–947
23. S. Jha, L. Kruger, P. McDaniel, Privacy preserving clustering, in *European Symposium on Research in Computer Security* (Springer, 2005), pp. 397–417
24. I.T. Jolliffe, Principal component analysis and factor analysis, in *Principal component analysis* (Springer, 1986), pp. 115–128
25. J. Konečnỳ, H.B. McMahan, F.X. Yu, P. Richtarik, A.T. Suresh, D. Bacon, Federated learning: Strategies for improving communication efficiency, in *NIPS Workshop on Private Multi-Party Machine Learning* (2016)
26. Q. Li, Z. Wen, Z. Wu, S. Hu, N. Wang, B. He, A survey on federated learning systems: vision, hype and reality for data privacy and protection (2019). arXiv:1907.09693
27. T. Li, A.K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, V. Smith, Federated optimization in heterogeneous networks. Proc. Mach. Learn. Syst. **2**, 429–450 (2020)
28. X. Li, M. Chen, F. Nie, Q. Wang, Locality adaptive discriminant analysis, in *Proceedings of the 26th International Joint Conference on Artificial Intelligence* (AAAI Press, 2017), pp. 2201–2207
29. J. Liu, C. Wang, J. Gao, J. Han, Multi-view clustering via joint nonnegative matrix factorization, in *Proceedings of the 2013 SIAM International Conference on Data Mining* (2013), pp. 252–260
30. D. Mascalzoni, A. Paradiso, M. Hansson, Rare disease research: breaking the privacy barrier. Appl. Transl. Genom. **3**(2), 23–29 (2014)
31. H.B. McMahan, E. Moore, D. Ramage, S. Hampson, et al., Communication-efficient learning of deep networks from decentralized data (2016). arXiv:1602.05629
32. A. Mizoguchi, A. Imakura, T. Sakurai, Application of data collaboration analysis to distributed data with misaligned features. Inf. Med. Unlocked **32**, 101013 (2022)
33. M.E.J. Newman, Finding community structure in networks using the eigenvectors of matrices. Phys. Rev. E **74**, 036104 (2006). https://doi.org/10.1103/PhysRevE.74.036104
34. A. Ng, M. Jordan, Y. Weiss, On spectral clustering: analysis and an algorithm. Adv. Neural Inf. Proc. Syst. **14**, 849–856 (2001)
35. H. Nguyen, D. Zhuang, P.-Y. Wu, M. Chang, Autogan-based dimension reduction for privacy preservation. Neurocomputing **384**, 94–103 (2020)
36. X. Ni, X. Shen, H. Zhao, Federated optimization via knowledge codistillation. Expert Syst. Appl. **191**, 116310 (2022)

37. F. Nie, J. Li, X. Li, et al., Parameter-free auto-weighted multiple graph learning: a framework for multiview clustering and semi-supervised classification, in *Proceedings of the 25th International Joint Conference on Artificial Intelligence* (2016), pp. 1881–1887
38. H. Park, A parallel algorithm for the unbalanced orthogonal procrustes problem. Parallel Comput. **17**(8), 913–923 (1991)
39. K. Pearson, LIII. On lines and planes of closest fit to systems of points in space. London, Edinburgh Dublin Philos. Mag. J. Sci. **2**(11), 559–572 (1901)
40. T. Sakurai, Y. Futamura, A. Imakura, T. Imamura, Scalable eigen-analysis engine for large-scale eigenvalue problems, in *Advanced Software Technologies for Post-Peta Scale Computing: The Japanese Post-Peta CREST Research Project* (Springer, 2019), pp. 37–57
41. T. Sakurai, H. Sugiura, A projection method for generalized eigenvalue problems using numerical integration. J. Comput. Appl. Math. **159**(1), 119–128 (2003)
42. B. Schölkopf, A. Smola, K.-R. Müller, Nonlinear component analysis as a kernel eigenvalue problem. Neural Comput. **10**(5), 1299–1319 (1998)
43. M. Sugiyama, Dimensionality reduction of multimodal labeled data by local Fisher discriminant analysis. J. Mach. Learn. Res. **8**, 1027–1061 (2007)
44. M. Sugiyama, T. Idé, S. Nakajima, J. Sese, Semi-supervised local Fisher discriminant analysis for dimensionality reduction. Mach. Learn. **78**, 35–61 (2010)
45. U. von Luxburg, A tutorial on spectral clustering. Stat. Comput. **17**(4), 395–416 (2007)
46. Q. Yang, Y. Liu, T. Chen, Y. Tong, Federated machine learning: concept and applications. ACM Trans. Intell. Syst. Technol. **10**(2), Article 12 (2019)
47. Y. Yang, H. Wang, Multi-view clustering: a survey. Big Data Min. Anal. **1**(2), 83–107 (2018)
48. T. Yano, Y. Futamura, A. Imakura, T. Sakurai, Efficient implementation of a dimensionality reduction method using a complex moment-based subspace, in *The International Conference on High Performance Computing in Asia-Pacific Region* (2021), pp. 83–89
49. X. Ye, H. Li, A. Imakura, T. Sakurai, Distributed collaborative feature selection based on intermediate representation, in *Proceedings of the 28th International Joint Conference on Artificial Intelligence* (2019), pp. 4142–4149
50. J. Zalonis, F. Armknecht, B. Grohmann, M. Koch, Report: state of the art solutions for privacy preserving machine learning in the medical context (2022). arXiv:2201.11406
51. P. Zhang, Y. Yang, B. Peng, M. He, Multi-view clustering algorithm based on variable weight and MKL, in *Proceedings of the International Joint Conference on Rough Sets* (2017), pp. 599–610

Chapter 5
Application of Tensor Network Formalism for Processing Tensor Data

Kenji Harada, Hiroaki Matsueda, and Tsuyoshi Okubo

5.1 Introduction

The growth of new mobility services, such as automatic driving and ride-sharing, requires a huge amount of data, for example, sensor images and information about the movement of various objects. These data often have multiple attributes, and are stored as a multi-dimensional array called a tensor. When the number of attributes (indexes) of a tensor increases, the size of the tensor becomes huge because it increases exponentially as the number of indexes increases. Therefore, the growth of future mobility services depends on the high-speed and high-quality use of tensor data of huge sizes.

A tensor network (TN) is an effective tool for representing a huge composite tensor and has been extensively developed in quantum information and statistical physics research [2, 12, 13]. For example, it has been used to represent a ground state in quantum many-body systems, where the dimension of the tensor is the number of quantum objects as qubits [13]. Since the tensor network can effectively represent such high-dimensional data, we may efficiently process the general tensor data. Recently, various applications of TN for general tensor data processing have been proposed [1, 5, 7, 10, 14, 18, 24].

In this chapter, we introduce three research areas in statistical physics that use tensor-network formalism to process tensor data: tree tensor networks (TTNs), tensor ring decomposition, and MERA, an extended tree tensor network. After briefly

K. Harada (✉)
Kyoto University, Yoshida-honmachi, Sakyo-ku, Kyoto 606-8501, Japan
e-mail: harada.kenji.8e@kyoto-u.ac.jp

H. Matsueda
Tohoku University, 6-6-05 Aramaki Aza Aoba, Aoba-ku, Sendai, Miyagi 980-8579, Japan
e-mail: hiroaki.matsueda.c8@tohoku.ac.jp

T. Okubo
University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-0033, Japan
e-mail: t-okubo@phys.s.u-tokyo.ac.jp

K. Ikeda et al. (eds.), *Advanced Mathematical Science for Mobility Society*,
https://doi.org/10.1007/978-981-99-9772-5_5

reviewing tensor-network formalism in statistical physics in Sect. 5.2, we explain generative modeling using a TTN for a multi-dimensional probability distribution [1, 5]. In Sect. 5.3, we introduce a new optimization algorithm for the network structure of a TTN. In Sect. 5.4, we outline tensor ring decomposition [24] for the compression of tensor data and explain our new approach for removing redundant information in tensor ring decomposition. In Sect. 5.5, we introduce an extended tree tensor network called MERA [21], which represents the compression of quantum information. We consider the underlying mechanisms of the success of MERA through the MERA representation of the ground state of a one-dimensional quantum model. Finally, in Sect. 5.6, we summarize these tensor-network approaches for tensor data processing.

5.2 Tensor-Network Formalism

5.2.1 Tensor Contraction and Tensor Network

Statistical causality between random variables is defined by conditional probability. For example, consider five random variables, x_1, x_2, x_3, x_4, and x_5, and assume that x_4 and x_5 statistically depend on (x_1, x_2) and (x_3, x_4), respectively. If all these random variables are discrete, and the support is finite, then the conditional probability $p(x_5|x_1, x_2, x_3)$ is defined by two conditional probabilities:

$$p(x_5|x_1, x_2, x_3) = \sum_{x_4} p_b(x_5|x_3, x_4) p_a(x_4|x_1, x_2), \tag{5.1}$$

where $p_a(|)$ and $p_b(|)$ are conditional probabilities. If we define the elements of a tensor as a conditional probability, that is,

$$\begin{aligned} A_{x_1x_2x_3} &\equiv p_a(x_3|x_1, x_2), \;\; B_{x_3x_4x_5} \equiv p_b(x_5|x_3, x_4), \\ T_{x_1x_2x_3x_5} &\equiv p(x_5|x_1, x_2, x_3), \end{aligned}$$

then (5.1) can be rewritten as

$$T_{x_1x_2x_3x_5} = \sum_{x_4} B_{x_3x_4x_5} A_{x_1x_2x_4}. \tag{5.2}$$

The right-hand side of (5.2) is an example of a tensor contraction, which is a generalization of a matrix product. A tensor contraction is defined as the summation of the multiplication of two tensor elements, with some indexes common to both tensors: for example, x_4 of A and B in (5.2). The composite tensor T is defined by the tensor contraction of A and B for the index x_4 in (5.2).

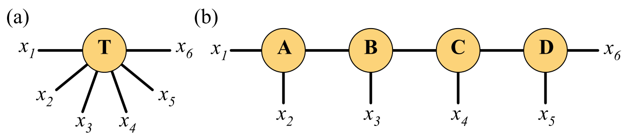


Fig. 5.1 Diagram notation of **a** the composite tensor T in (5.3); **b** the tensor contractions in (5.3)

Using tensor contractions, we can define a large tensor T as

$$T_{x_1x_2x_3x_4x_5x_6} = \sum_{y_1}\sum_{y_2}\sum_{y_3} A_{x_1x_2y_1} B_{y_1x_3y_2} C_{y_2x_4y_3} D_{y_3x_5x_6}. \tag{5.3}$$

Visually understanding the relationship of each tensor in tensor contractions using only (5.3) is difficult; a diagram notation is useful for visualizing this. Figure 5.1a and b are diagram notations for the left-hand and right-hand sides of (5.3), respectively. As shown in Fig. 5.1a, a geometrical object (circle) is used to represent a tensor T. The open edges represent the indexes of the tensor. In Fig. 5.1b, an edge between two tensors denotes a tensor contraction. A connected edge indicates the connection of two edges (indexes) for a tensor contraction. In summary, a node and connection of nodes in these diagrams represent a tensor and tensor contraction, respectively. The diagram in Fig. 5.1b represents a composite tensor as a network of tensors; hence, we call this a tensor network.

Since a tensor network defined by a diagram is a composite tensor, we can define the class of composite tensors as a tensor network. Several types of network structures for tensor networks have already been proposed. For example, a tensor network with the one-dimensional structure shown in Fig. 5.1b is called a matrix product state (MPS). Historically, tensor networks have been used in the field of quantum information. If quantum amplitudes in a wave function are defined by a tensor, the tensor can be mapped to a quantum state. For example,

$$T \leftrightarrow |\psi\rangle = \sum_{x_1,x_2,x_3} T_{x_1x_2x_3}|x_1x_2x_3\rangle, \tag{5.4}$$

where $|x_1x_2x_3\rangle$ is a base, that is, a tensor product of local bases in local Hilbert spaces. Thus, tensor networks can be called quantum states. In addition, an MPS is called a tensor train (TT) in applied mathematics [14]. Generally, if all tensors have open edges and are connected to their neighboring tensors, the tensor network is called a tensor product state (TPS) [9]. The one-dimensional TPS is the MPS, and the two-dimensional TPS is called a projected entangled paired state (PEPS) [20]. A tensor network without a loop structure is called a tree, as shown in Fig. 5.2, and various tree tensor networks (TTNs) exist. Note that an MPS belongs to the TTN class.

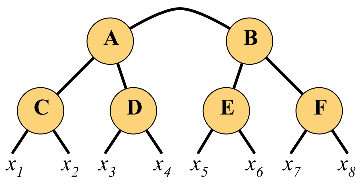


Fig. 5.2 Diagram notation of a tree tensor network

5.2.2 Tensor Decomposition and Tensor Compression

The singular value decomposition of a $m \times n$ matrix A is $U\Lambda V^\dagger$, where U and V are isometries, as $U^\dagger U = V^\dagger V = I$, and Λ is a positive diagonal matrix. The diagonal part of Λ consists of positive singular values. We can perform SVD for any matrix. If the number of singular values is smaller than m and n, or the number of singular values is reduced by removing small values, then the matrix A can be compressed. Although the number of elements in A is $m \times n$, the total number of independent elements in U, V, and Λ is $(m + n - k)k$. Therefore, if $k < \min(m, n)$, the SVD is a compression of the original matrix. The SVD with the k largest singular values is the best approximation of A within the rank-k matrix: $\tilde{U}\tilde{\Lambda}\tilde{V}^\dagger = \arg\min_{\tilde{A}:\text{rank}-k} |\tilde{A} - A|_F$, where $|X|_F$ is the Frobenius norm, $|X|_F \equiv \text{Tr}[X^\dagger X]$.

Tensor decomposition is the transformation of a tensor into a composite tensor. Since composite tensors can be defined as a tensor network, a tensor network represents a tensor decomposition. For example, we can regard a tensor as a matrix by splitting the indexes of the tensor into two groups and compositing the indexes in each group. Thus, using SVD, we can decompose a matrix into a product of two matrixes: $T = U\Lambda V^\dagger = (U\sqrt{\Lambda})(\sqrt{\Lambda}V^\dagger) = AT'$. By decomposing composite indexes into original indexes, we can transform the original tensor into a tensor contraction of two tensors. Repeatedly applying this decomposition to the right-hand side, T', we can transform any tensor into an MPS without truncations, as shown in Fig. 5.1b. Therefore, an MPS (TT) is a tensor decomposition that is applicable to any tensor. If the dimension of each tensor index is m and the number of indexes is n, then the number of elements in T is m^n. However, the number of elements in the MPS is $O(n \times (\chi^2 m))$, where χ is the dimension of the indexes between neighboring tensors in the MPS, called a bond dimension. Therefore, an MPS with a fixed bond dimension yields great compression.

If the largest singular values are kept, the approximation can be controlled by the MPS. However, determining the best tensor network with a fixed number of parameters to approximate a given tensor is difficult. Thus, we use a tensor network as a variational ansatz.

The computational cost of a tensor contraction is the product of all dimensions of the indexes in the tensor network that represent the tensor contraction. The computational cost of a tensor network that contains many tensor contractions generally depends on the order in which the tensor contractions are processed; however, determining the best order is an NP-hard problem. In addition, even for a compact tensor network such as a PEPS, the computational cost and spatial one exponentially increase with the number of tensors. Therefore, various approximation methods for tensor contractions have been proposed.

Using a tensor-network structure, we can efficiently calculate tensor contractions in some cases. For example, the computational cost of the inner product of two MPSs is only proportional to the length of the MPS. Therefore, compressing a huge tensor using a proper tensor-network decomposition can significantly reduce the computational cost of processing the tensor data.

5.3 Generative Model Using a Tree Tensor Network

5.3.1 Generative Modeling

Generative modeling finds a parametrized distribution $p(\mathbf{x})$ to approximate a data distribution $\pi(\mathbf{x})$ [16]. Since the distance between these distributions can be defined by the Kullback–Leibler (KL) divergence,

$$D_{\mathrm{KL}}(\pi||p) = \sum_{\mathbf{x}} \pi(\mathbf{x}) \ln\left(\frac{\pi(\mathbf{x})}{p(\mathbf{x})}\right), \tag{5.5}$$

we can optimize $p(\mathbf{x})$ to minimize this KL divergence.

Suppose that a data sample is a vector in which each element takes a state from a finite set of states: $\{\mathbf{x}|(\mathbf{x})_i \in \{1, 2, \dots, m\}\}$. A set of data samples, $\mathcal{M} = \{\mathbf{x}_\mu\}_{\mu=1,\dots,M}$, defines an empirical distribution:

$$\pi(\mathbf{x}) = \frac{1}{M}\sum_{\mu=1}^{M} \delta(\mathbf{x}, \mathbf{x}_\mu), \tag{5.6}$$

where M is the number of data samples and $\delta(\mathbf{x}, \mathbf{y})$ is one if $\mathbf{x} = \mathbf{y}$, otherwise it is zero. In practice, when the target distribution of generative modeling is the empirical data distribution, we minimize the negative log-likelihood (NLL) as the loss function during learning,

$$\mathcal{L} = -\frac{1}{M}\sum_{\mathbf{x}\in\mathcal{M}} \ln[p(\mathbf{x})] = S(\pi) + D_{\mathrm{KL}}(\pi||p), \tag{5.7}$$

where $S(\pi)$ is the entropy of the π distribution.

5.3.2 Tree Generative Model

Here, we consider a generative model based on a quantum state [1, 5]. Following the Born rule, we define $p(\mathbf{x})$ as the square of the amplitude of a wave function:

$$p(\mathbf{x}) = \frac{|\psi(\mathbf{x})|^2}{Z}, \tag{5.8}$$

where $\psi(\mathbf{x})$ is a wave function and Z is the normalization factor.

MPSs [5] and TTNs [1] have been proposed to define the wave function for generative modeling. Figure 5.1b shows the network structure of an MPS. Each tensor in the MPS has three edges, and these edges are sequentially connected. Figure 5.2 shows the network structure of a TTN. The number of indexes of a tensor in this TTN is equal to that in the MPS. The only difference is the topology of the network; all physical indexes in the TTN, x_i, are connected, and the TTN network has no loop structure. Thus, an MPS is a specialized type of TTN. In the following, a generative model using a TTN is called a tree generative model.

5.3.3 Canonical Form of TTN

Using redundancy to insert a pair of matrices and their inverses on an edge in a tensor network, we can construct a useful canonical form of a TTN [17]. Since a TTN has no loop, we can decompose the network of a TTN into two trees by cutting an edge, as shown in Fig. 5.3a. If we regard the cut edge as the root edge of each tree, we can define the order of nodes from the terminal nodes (leaves) to the root and apply the following tensor transformations in this order. Each tensor has an edge toward the root node and the remaining two edges. By combining these two edges into an index, the tensor is transformed into a matrix: $T_{ijk} = M_{i(jk)}$. The SVD of the matrix splits the matrix into an isometry and a matrix connected to the edge toward the root node (see Fig. 5.3b):

$$M_{i(jk)} = \sum_l M'_{il} U_{l,(jk)}, \tag{5.9}$$

where U is an isometry as

$$\sum_{(jk)} U_{l,(jk)} U_{l',(jk)} = \delta_{l,l'}. \tag{5.10}$$

The matrix M' is then absorbed by the next tensor (see Fig. 5.3c). This procedure is repeated from the leaf tensors to the root node. Then, two modified root tensors are obtained and combined into a tensor with four edges. Finally, the SVD of the

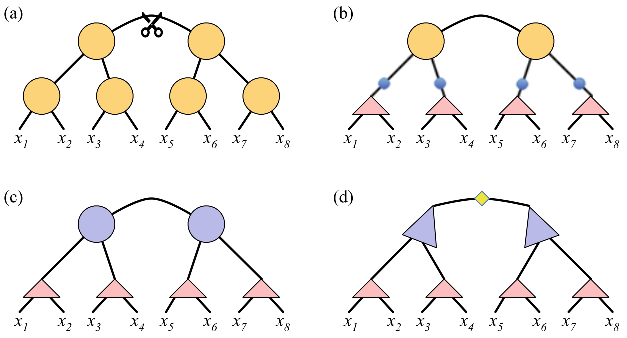


Fig. 5.3 Constructing the canonical form of a TTN: **a** cutting an edge of the TTN; **b** splitting a tensor into an isometry (triangle) and a matrix (circle) using the SVD of a leaf; **c** matrices from the SVD are absorbed into an upper tensor; and **d** the canonical form of the TTN (the rhombus represents a diagonal matrix that consists of singular values)

top tensor obtains two isometries and a diagonal matrix that consists of the singular values (see Fig. 5.3d).

The canonical form of a TTN is useful as it enables a direct calculation of the normalization factor in (5.8). Since almost all tensors in the canonical form are isometries with the property defined in (5.10), the normalization factor directly depends on the singular values of the canonical form of the TTN, $\{\lambda_i\}$:

$$Z = \sum_{\mathbf{x}} |\psi(\mathbf{x})|^2 = \sum_{\mathbf{x}} \langle \psi(\mathbf{x}) | \psi(\mathbf{x}) \rangle = \sum_i (\lambda_i)^2. \tag{5.11}$$

To calculate the NLL, we need to estimate a set of quantum amplitudes for the data, useful for the network structure of a TTN. The data vector $\mathbf{x}_\alpha$ of a sample α can be decomposed to a direct product of local vectors: $(\mathbf{x}_\alpha)^{(1)} \otimes (\mathbf{x}_\alpha)^{(2)} \otimes \cdots$. The set of local vectors for samples at a site i is represented by the matrix $V^{(i)}_{k\alpha}$, where k is an index of the data at a site i. Thus, the total data set is defined as a TTN defined by the same network structure of delta tensors, where the leaves are the matrices $V^{(i)}$, as shown in Fig. 5.4a; the delta tensor is one if all indexes are the same, otherwise it is zero. Due to the network structure of a TTN, we can efficiently calculate the contraction of the TTN with a data TTN using recursive steps (see Fig. 5.4a–c.)

.

5.3.4 *Learning Algorithm*

Previous studies [1, 5] have used the learning algorithm for a single node (tensor) near the center of a canonical form with singular values, for example, the part enclosed by the dotted line in Fig. 5.4a. After combining the isometry and singular values in

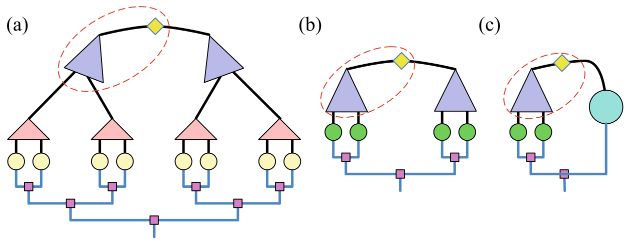


Fig. 5.4 Evaluation of quantum amplitude: **a** quantum amplitude $\psi(\mathbf{x}_\alpha)$ for a sample α; and **b** and **c** recursive calculation of $\psi(\mathbf{x}_\alpha)$, except for the part enclosed by the dotted line. The circles indicate data matrices at site i and the squares indicate a delta tensor, which is one if all indexes are the same, otherwise, it is zero

the enclosed part into a single tensor (node) with 3-legs, it can be updated using the gradient of the NLL. The SVD of the update tensor into an isometry, singular values, and a unitary can recover the canonical form of the TTN. Since the center of the canonical form of a TTN can move to a neighboring position in the network using SVD [17], we sweep all nodes in a TTN with single node updates.

5.3.5 *Network Optimization*

The network structure of a TTN is important for generative modeling. The performance of the balance tree generative model is better than that of the MPS in several scenarios [1]. Both the MPS and balance tree belong to the tree network class. The difference between them is the structure of their network. Therefore, optimizing the network for given data is effective for generative modeling.

The DMRG algorithm [22, 23] is often used to find the ground state of a one-dimensional quantum model, which is a variational method for an MPS. For a finite system, the DMRG algorithm sweeps and optimizes the tensors in the canonical form of an MPS to improve the variational energy for a target Hamiltonian. A two-site DMRG algorithm simultaneously updates two neighboring tensors in an MPS. Two tensors that are directly connected can be combined into a 4-leg tensor, and this combined tensor can be updated. Finally, the combined tensor is decomposed into two 3-leg tensors using SVD, and the algorithm proceeds to the next pair. The corresponding algorithm for a tree generative model was proposed in [1]. In these studies[1, 22, 23], the network structure of the TTN does not change. The 4-leg tensor can be divided into two tensors in three possible ways, as shown in Fig. 5.5. Selecting a new division globally changes the network structure of the TTN.

We now propose a new algorithm to change the network structure of a TTN for generative modeling. First, two isometries and singular values are combined in the center of the canonical form into a 4-leg tensor. The 4-leg tensor is updated to improve the NLL and is then converted into a matrix. Three matrices are used to divide the

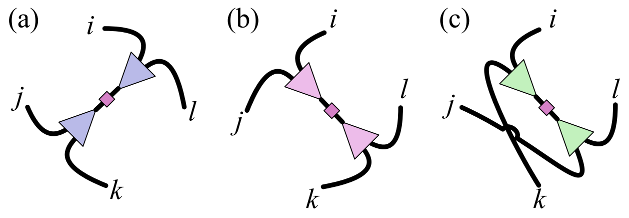


Fig. 5.5 Decomposition of a 4-leg tensor. The triangles and squares represent isometries and diagonal matrices of singular values, respectively

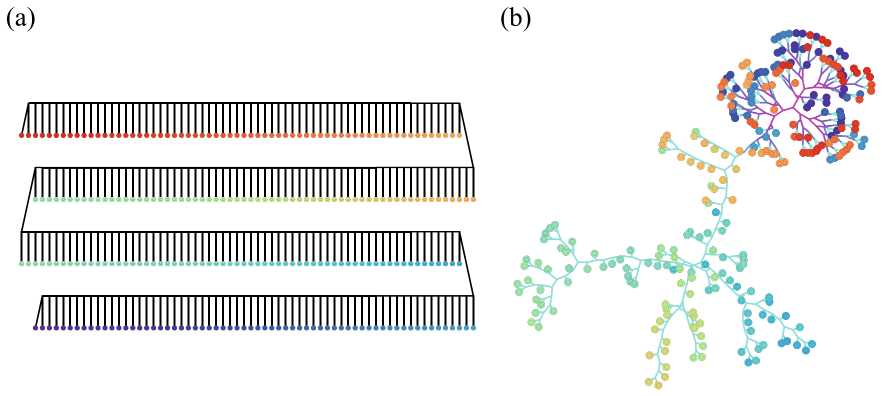


Fig. 5.6 Network structure of a tree generative model: **a** initial MPS structure for random patterns and **b** network structure after optimization by our proposed algorithm. Circles indicate probability variables and their color indicates their position in the line. Edges indicate the index of a tensor, which is a vertex with three legs

4-leg tensor into two groups. A better division is selected and the SVD of the matrix is conducted to produce a new canonical form. The center of the canonical form is moved, and the updates are repeated. Several strategies can be used to select a better division; further details can be found in [6].

We tested the proposed algorithm for the empirical probability distribution of 10 random patterns, starting from the MPS structure shown in Fig. 5.6a, because the binary probability variable is on a line. The length of the line is 256. We fixed the center part of the line at 0 and divided the left- and right-hand sides into a random pattern, as shown in the top and bottom rows in Fig. 5.6a. The variables on the left-hand side strongly correlate to those on the right. In this case, when we start with a randomly initialized MPS, we cannot obtain the minimum NLL if the network structure is fixed as an MPS. However, we can find a tree generative model with the minimum NLL by changing the structure of the network using our proposed algorithm. Figure 5.6b shows the optimized structure of the network using the proposed algorithm; this interesting network structure spontaneously emerged. Since the probability variables of the left- and right-hand sides strongly correlate, they are embedded into a compact tree structure, shown in the upper right of Fig. 5.6b. However, the lower part of Fig. 5.6b consists of probability variables in the center, which are fixed to 0.

5.4 Tensor Ring Decomposition

5.4.1 Introduction to Tensor Ring Decompositions

As discussed in Sect. 5.2, we can consider a variety of tensor-network decompositions to represent a given tensor. When we perform approximation for such a tensor network decomposition, the efficiency of the data compression highly depends on the structure of the tensor-network and properties of the tensor data. For quantum many-body problems, the area law of entanglement entropy, which determines the scaling of the amount of correlation in the quantum state, plays an important role in selecting a better tensor network. However, the area law does not necessarily hold for general tensor data. Thus, it remains unclear which types of tensor networks, beyond the simplest form of MPS (TT), are suitable for decomposing general tensor data.

In this section, we consider tensor ring decomposition (TRD), which is a fundamental decomposition of multidimensional tensors [24]. In TRD, tensors are decomposed into a form of matrix product states with periodic boundary conditions (Fig. 5.7). For example, for the N-leg tensor $T_{i_1,i_2,\ldots,i_N}$, the TRD is expressed as

$$\begin{aligned} T_{i_1,i_2,\ldots,i_N} &= \sum_{j_1,j_2,j_3} M^{(1)}_{j_1,j_2}[i_1] M^{(2)}_{j_2,j_3}[i_2] \cdots M^{(N)}_{j_N,j_1}[i_N] \\ &= \mathrm{Tr} \prod_{n=1}^{N} M^{(n)}[i_n], \end{aligned} \tag{5.12}$$

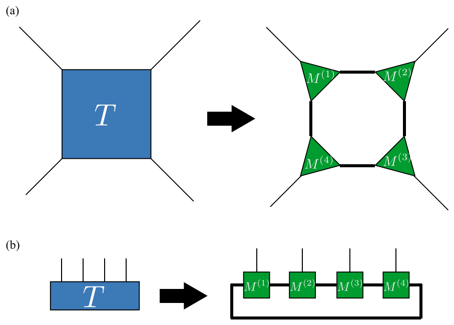


Fig. 5.7 Tensor ring decomposition (TRD) of a 4-leg tensor T into four 3-leg tensors, $M^{(i)}$: **a** a symmetric TRD diagram and **b** an alternative representation of the TRD diagram

where $M^{(n)}_{j_n,j_{n+1}}[i_n]$ is a 3-leg tensor, $i_n = 1, 2, \ldots, d_n$, and $j_n = 1, 2, \ldots, D_n$. Note that in the last expression, $M^{(n)}[i_n]$ is regarded as a matrix for a fixed i_n. Hereafter, we denote a tensor defined by (5.12) as $\mathrm{tTr} \prod_{n=1}^{N} M^{(n)}$, where tTr indicates the trace of a tensor network. We can obtain a TRD of a certain tensor, for example, by using successive SVDs; however, this yields an (open boundary) MPS, i.e., the bond dimensions of the two boundaries, D_1 and D_N, become one. However, an open boundary MPS solution is not usually optimal if we want to minimize the maximum or average bond dimension in the TRD, because information needs to flow through the entire system to represent correlations between two edges, which increases the bond dimension, D_n. Thus, an efficient algorithm is required to find the optimal TRD for given tensors.

We can also consider a related problem where we instead approximate an N-leg tensor using TRD for given bond dimensions D_n to find the exact TRD. The alternated least square (ALS) method [24] is often used to find a numerically approximate TRD. The aim of the ALS method is to find a TRD that minimizes the distance between the original tensor and a TRD representation defined by the Frobenius norm:

$$F = \left\| T - \mathrm{tTr} \prod_{n} M^{(n)} \right\|^2 . \tag{5.13}$$

The optimal solution is iteratively searched for by solving local linear problems for $M^{(n)}$ defined by fixing $M^{(m)}$ for $m \neq n$. When a 3-leg tensor $M^{(n)}$ is regarded as a vector,

$$(\mathbf{M})_{(j_1,j_2,i)} = M^{(n)}_{j_1,j_2}[i], \tag{5.14}$$

the squared norm F can be written as

$$F = \|T\|^2 + \mathbf{M}^\dagger \hat{N} \mathbf{M} - \mathbf{M}^\dagger \mathbf{W} - \mathbf{W}^\dagger \mathbf{M}, \tag{5.15}$$

where $\hat{N}$ and $\mathbf{W}$ are defined as in Fig. 5.8.

Note that the matrix N is positive-semidefinite by construction. Because F is a quadratic function of $\mathbf{M}$, its extrema are given by solving the linear equations defined by

$$\hat{N}\mathbf{M} = \mathbf{W}. \tag{5.16}$$

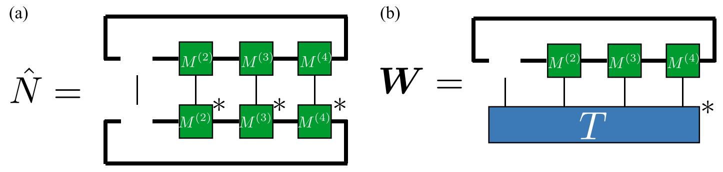


Fig. 5.8 Matrix $\hat{N}$ and vector $\mathbf{W}$ in (5.15) for $n = 1$

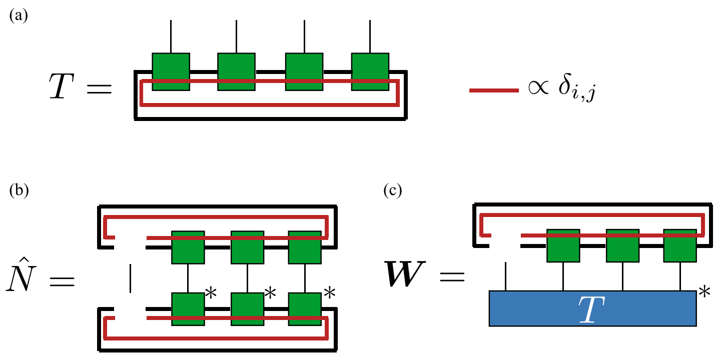


Fig. 5.9 TRD that contains a redundant loop: **a** the TRD of a tensor T, where the red line indicates the redundant loop; **b** the corresponding matrix $\hat{N}$; and **c** the vector $\mathbf{W}$ for $n = 1$

Thus, when the matrix $\hat{N}$ is a regular matrix, we can obtain the optimal $\mathbf{M}$:

$$\mathbf{M} = \hat{N}^{-1}\mathbf{W}. \tag{5.17}$$

However, in general, $\hat{N}$ can have zero eigenvalues and its inverse matrix $\hat{N}^{-1}$ may not exist. In this case, the linear problem becomes underdetermined. To solve this problem, the pseudo inverse (PI), $\hat{N}^{+}$, is often used instead of the inverse. Alternatively, the conjugate gradient (CG) method can be used to obtain one of the solutions.

The ALS algorithm often gets trapped at a local minimum of F. In particular, if the initial estimate of $M^{(n)}$ contains a "redundant loop", removing such a contribution from the TRD is not easy; thus, it will not converge to the global minimum. We investigate such a situation in Sect. 5.4.2.

5.4.2 *Redundant Loops*

We now consider the numerically optimized TRD of T, starting from an initial estimate that includes a redundant loop. For a general TRD, tTr $\prod_n M^{(n)}$, we define an ideal redundant loop by adding additional degrees of freedom in the virtual bond for all $M^{(n)}$:

$$\tilde{M}^{(n)}_{(j_n,k_n),(j_{n+1},k_{n+1})}[i_n] \equiv \sigma^{(n)}_{k_n}\delta_{k_n,k_{n+1}}M^{(n)}_{j_n,j_{n+1}}[i_n]. \tag{5.18}$$

This is also shown in Fig. 5.9a. Clearly, the TRD represented by $\tilde{M}^{(n)}$ is equivalent to that of $M^{(n)}$ up to a constant $\sum_k \prod_n \sigma^{(n)}_k$. Thus, no essential information is represented by the additional indices k_n.

When a TRD has redundant loops, the matrix $\hat{N}$ for $n = 1$ is represented as in Fig. 5.9b. The redundant loops in the upper and bottom parts of this figure are disconnected, indicating that $\hat{N}$ has zero eigenvalues.

Similarly, the vector $\mathbf{W}$ for $n = 1$ is represented as in Fig. 5.9c. We can easily confirm that the solution obtained by the pseudo inverse of N, $\mathbf{M} = \hat{N}^{+}\mathbf{W}$, maintains the redundant loop. However, the redundant loop can disappear in the general solution of the linear equation along with the eigenvectors that correspond to the zero eigenvalues. Thus, to remove redundant loops, we must properly determine the contribution of the zero eigenvectors.

5.4.3 Entanglement Penalty Algorithm

We now discuss an idea to improve the optimization of the TRD, inspired by quantum many-body problems. We consider the corner double line (CDL) tensor, which often appears in statistical physics [4], as a model tensor. Based on the CDL structure, we introduce a modified cost function that can avoid the previously discussed local minima.

Let a tensor T be represented by an exact TRD:

$$T_{i_1,i_2,\ldots,i_N} = \mathrm{Tr} \prod_{n=1}^{N} C^{(n)}[i_n]. \tag{5.19}$$

Assume that each index i_n is represented by a set of two indices (x_n, y_n) and that the 3-leg tensor $C^{(n)}$ has a CDL structure:

$$C^{(n)}_{j_n,j_{n+1}}[i_n = (x_n, y_n)] = \lambda^{(n)}_{j_n} \delta_{j_n,x_n} \delta_{j_{n+1},y_n}. \tag{5.20}$$

T and $C^{(n)}$ can be represented as in Fig. 5.10. More generally, we can consider unitary matrices that mix the indices x_n and y_n. In this case, $C^{(n)}$ is written as

$$C^{(n)}_{j_n,j_{n+1}}[i_n = (x_n, y_n)] = \lambda^{(n)}_{j_n} U^{(n)}_{(j_n,j_{n+1}),(x_n,y_n)}, \tag{5.21}$$

where $U^{(n)}$ is a unitary matrix.

As discussed in Sect. 5.4.2, when the ALS algorithm is used to find an optimal TRD of T, represented by (5.19), from an initial guess that includes redundant loops, it gets trapped at a local minimum. To avoid such local minima, we consider an additional term in the cost function.

One of the differences between the redundant loop and CDL structure is the *entanglement*, or correlation, between the original indices, i_n, and virtual indices in the TRD. For redundant loops, the original and virtual indices have no connection, but in the CDL structure they do. When we regard a 3-leg tensor M_{i,j_1,j_2} as a matrix $M_{i,(k_1,k_2)}$, the amount of such entanglement can be characterized by *entanglement entropy*, which is often used in quantum information [3] and is defined using the singular values of M:

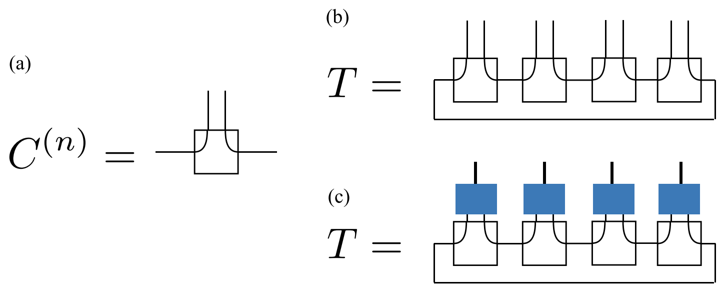


Fig. 5.10 Ideal corner double line (CDL) tensors: **a** a local CDL tensor; **b** a 4-leg tensor consisting of general CDL tensors without unitary transformations; and **c** a 4-leg tensor consisting of general CDL tensors with unitary transformations. The blue squares indicate unitary matrices

$$S_{\mathrm{E}} \equiv -\sum_i \tilde{s}_i \log \tilde{s}_i, \tag{5.22}$$

where $\tilde{s_i} = s_i / \sum_i s_i$ is the normalization of the singular value s_i. When M is an $N \times M$ matrix and $r = \min(M, N)$, S_{E} is such that $0 \leq S_{\mathrm{E}} \leq \log r$: $S_{\mathrm{E}} = 0$ for $\tilde{s}_1 = 1$ and $\tilde{s}_i = 0 (i \neq 1)$, and $S_{\mathrm{E}} = \log r$ for $\tilde{s}_i = 1/r$. Note that $S = 0$ corresponds to the redundant loop, and S_{E} takes larger finite values for the CDL. Thus, when we consider an additional term (negatively) proportional to the entropy in the cost function, CDL-like solutions may be favored over the redundant loop. The new cost function is explicitly written as

$$F'(\epsilon) = \left\| T - \mathrm{tTr} \prod_n M^{(n)} \right\|^2 - \epsilon \frac{1}{N} \sum_n S_{\mathrm{E}}(M^{(n)}), \tag{5.23}$$

where ϵ is a positive constant. For a finite ϵ, the global minimum of F' yields a different TRD from that which minimizes F. Thus, in practice, we may begin the ALS algorithm with a sufficiently large ϵ and adjust it toward $\epsilon = 0$ with each iteration. Using this procedure, the TRD may escape the local minimum and redundant loop during the initial iterations; then, when $\epsilon = 0$, the ALS algorithm is expected to converge to the global minimum of $F'(\epsilon = 0) = F$.

5.4.4 Numerical Experiments

We will now demonstrate how the entanglement entropy penalty works using numerical experiments. In these experiments, we performed ALS-like site-wise optimization with the cost functions F and $F'(\epsilon)$ to find the optimal TRD. Each iteration of the ALS algorithm involves updating N local 3-leg tensors. For F', the local problem

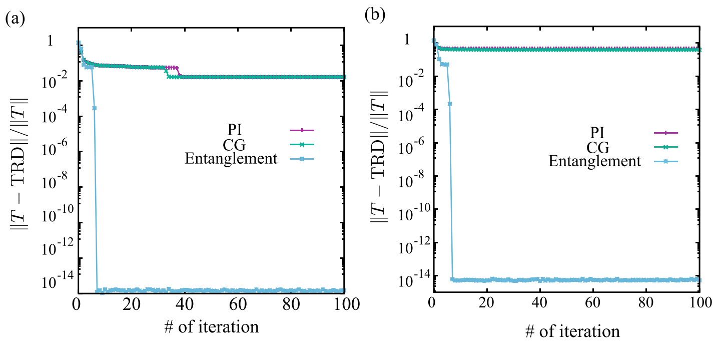


Fig. 5.11 Optimization of TRD using the standard cost function with the PI and CG solvers versus the entanglement cost function for a random 4-leg tensor consisting of CDL tensors with a bond dimension of 16 for the outer indices: **a** the initial tensors are random dense tensors and **b** the initial tensors are random dense tensors with redundant loops with a bond dimension of 2, as in (5.18). For the entanglement cost function, $\epsilon = 0.1$ for the first five iterations, which was minimized by the CG method; then, $\epsilon = 0$, which was minimized by the PI. Each iteration optimizes N tensors. Here, $N = 4$

becomes nonlinear; we typically use the CG method to solve such a problem. For F, we can use either the PI or CG method.

As the simplest example, we show the typical optimization dynamics for the ideal CDL represented by (5.21). Figure 5.11a shows the convergence of the ALS algorithm, starting from random tensors, for a 4-leg tensor T consisting of a CDL with a bond dimension of each index i_1 of 16. For the entanglement cost function (5.23), we set $\epsilon > 0$ for the first few steps and then $\epsilon = 0$, i.e., $F'(\epsilon = 0) = F$, to obtain the true global minimum. We use the PI and CG methods to minimize the standard ALS cost function (5.13) and use the CG method for the entanglement cost function. The optimization with the usual cost function fails to converge to the global minimum with the PI and CG methods. However, the entanglement cost function (5.23) converges to the global minimum. When we consider initial tensors with the ideal redundant loops defined in (5.18), the standard ALS almost always gets trapped at a local minimum, as shown in Fig. 5.11b. However, the entanglement cost function avoids getting trapped at local minima and smoothly converges to the global minimum. Further details and results of these numerical experiments are presented elsewhere [11].

These numerical experiments indicate that the entanglement cost function works very well for CDL-type tensors. However, in general, the target of TRD can be very different from the ideal CDL. Our other experiments using tensors constructed from the TRD of general tensors indicate that a naive entanglement cost function does not always escape local minima; it depends on the balance of the bond dimensions of the outer and virtual indices [11].

5.5 Exact MERA Network and Quantum Renormalization Group for Critical Spin Models

5.5.1 Introduction

In the previous sections of this chapter, we showed that tensor-network approaches are applicable to information-scientific problems. Thus, a deep understanding of these successes is highly desirable. In quantum physics, the approaches were originally proposed as efficient numerical methods for treating the ground and low-lying states of strongly interacting quantum many-body systems. In this section, we examine the underlying mechanisms of the success of a class of tensor networks called multiscale entanglement renormalization ansatz (MERA) [21].

Usually, two different types of tensor networks are considered depending on how close the target model is to quantum criticality. This is because the magnitude of non-local quantum correlation or entanglement determines the structure of the network. Away from criticality, the quantum state can be well represented by the PEPS. The well-known MPS is a one-dimensional realization of PEPS, and the relevant matrix dimension characterizes the magnitude of the quantum entanglement. Alternatively, the algebraic Bethe ansatz is a mathematically exact method for one-dimensional solvable models, and this can also be transformed into an MPS. Thus, this recent approach also has a traditional root, and hence the underlying mathematics is clear. However, in quantum critical cases, the ability of PEPS greatly reduces because the tensor dimension must be increased to precisely numerically optimize the tensor-network wave function. In this case, including an extra dimension, the so-called holographic dimension, in the network is a better approach. This extra dimension corresponds to the flow of real-space renormalization and also plays a role in greatly reducing the tensor dimension. The corresponding network is the MERA network.

Surprisingly, the MERA concept stimulates string theorists because the structure of the network is quite similar to the spacetime concept that appears in gauge/gravity correspondence [8, 15, 19]. This correspondence is considered to be key to understanding the complementary relationship between quantum field theory and general relativity. Therefore, clarifying the mathematical structure of the MERA network is important interdisciplinary research beyond condensed matter and statistical physics.

The traditional methods of a renormalization group (RG) are based on the flow of interaction parameters in the Hamiltonian, which is defined by the repetition of the renormalization group transformation in real or momentum space. By transforming the Hamiltonian, we can determine how the system approaches the fixed point and what the dominant parameters are. However, recent tensor-network approaches are mainly based on the optimization of the variational ansatz, and their relationship with the traditional RG concept is not very clear. We aim to overcome this discrepancy by bridging the tensor network with the RG of the Hamiltonian.

5.5.2 Heisenberg Model and Quantum Entanglement

We start with the antiferromagnetic Heisenberg Hamiltonian in one spatial dimension:

$$H = J \sum_{i=1}^{N} \mathbf{S}_i \cdot \mathbf{S}_{i+1}. \tag{5.24}$$

Here, $\mathbf{S}$ is a quantum spin operator, $\mathbf{S} = \frac{1}{2}\boldsymbol{\sigma}$ for Pauli matrix $\boldsymbol{\sigma}$, J is the exchange coupling, N is the number of lattice sites, and we assume the periodic boundary condition $\mathbf{S}_{N+1} = \mathbf{S}_1$. The ground state of this Hamiltonian is $|\psi\rangle = \sum_{s_1,\ldots,s_N} \psi^{s_1 \cdots s_N} |s_1 \ldots s_N\rangle$, where $|s_1 \ldots s_N\rangle$ is the abbreviation of $|s_1\rangle \otimes \cdots \otimes |s_N\rangle$.

As an example, we consider the 4-site case ($N = 4$), for which we can obtain the exact eigenstates (this is a very pedagogical example to demonstrate the nature of the MERA network). The eigenvalues are $E = -2J, -J, 0, 0, 0, J$. In particular, the ground state ($S^z_{tot} = 0$) is given by

$$\sqrt{12}\,|\psi\rangle = 2\left(|\uparrow\downarrow\uparrow\downarrow\rangle + |\downarrow\uparrow\downarrow\uparrow\rangle\right) - \left(|\uparrow\uparrow\downarrow\downarrow\rangle + |\uparrow\downarrow\downarrow\uparrow\rangle + |\downarrow\uparrow\uparrow\downarrow\rangle + |\downarrow\downarrow\uparrow\uparrow\rangle\right), \tag{5.25}$$

where the coefficient $\sqrt{12}$ corresponds to a normalization factor of $|\psi\rangle$. In addition to the first term on the right-hand side of the equation, which is a classical antiferromagnetic configuration, a second term exists due to quantum fluctuation. The second term represents domain excitation, $|\uparrow\downarrow\rangle_{i,i+1} \otimes |\downarrow\uparrow\rangle_{i+2,i+3}$. This state can also be written as

$$\sqrt{3}\,|\psi\rangle = |s\rangle_{12} \otimes |s\rangle_{34} + |s\rangle_{23} \otimes |s\rangle_{41}\,, \tag{5.26}$$

where $|s\rangle_{ij} = \left(|\uparrow\downarrow\rangle_{ij} - |\downarrow\uparrow\rangle_{ij}\right)/\sqrt{2}$. Thus, two singlets spatially fluctuate, and this state has a finite amount of quantum entanglement. To clarify this feature, we introduce a reduced density matrix by taking the partial trace of the density matrix $\rho = |\psi\rangle\langle\psi|$ as $\rho_{12} = tr_{34}\rho$. Then, the bipartite entanglement entropy is defined by

$$S_{12} = -tr_{12}\left(\rho_{12}\log\rho_{12}\right) = S_{12} = 2\log 2 - \frac{1}{2}\log 3 = 0.83698\cdots. \tag{5.27}$$

Because of the finite S_{12}, simple treatments, such as mean-field theory, break down. Tensor-network methods can efficiently treat this entanglement. For a one-dimensional case, this model can be solved exactly, even for a general N. In this case, the wave function is represented by the Bethe ansatz. The algebraic version of this ansatz can be transformed into the matrix product state.

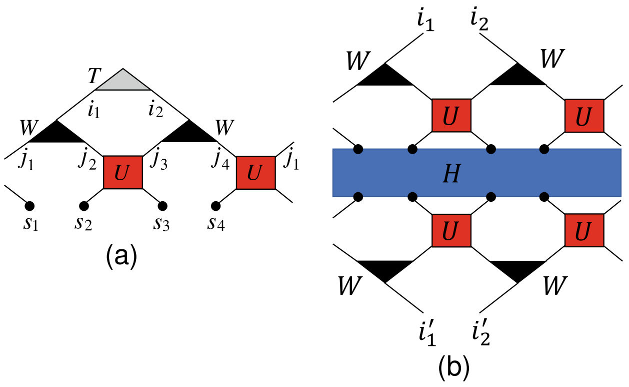


Fig. 5.12 **a** MERA tensor network ($N = 4$, periodic boundary condition) and **b** graphical representation of the effective two-site Hamiltonian after the quantum RG processes represented by disentangling and isometry. Here, the two isometries correspond to the effective sites

5.5.3 Construction of Exact MERA Network

We would like to represent $|\psi\rangle$ by the hierarchical tensor network (MERA) in Fig. 5.12. The hierarchical network matches the quantum critical systems since it can represent the presence of various energy scales due to the real-space renormalization processes. This construction can greatly reduce the tensor dimension compared with the MPS representation. This is advantageous in terms of numerical simulation. Furthermore, the hierarchical network contains disentangling tensors. The presence of disentangling tensors is necessary to realize the success of the real-space RG in quantum systems. Because of quantum fluctuation, the spin correlation is essentially nonlocal, thus maintaining a good approximation using only local transformations is difficult. Before the RG, the disentangling tensors properly kill the nonlocal entanglement, and thus the real-space RG becomes successful. This is a key in MERA optimization. Note that such a network without the disentangling tensors is a tree tensor network. The tree tensor network has various applications, as presented earlier in this chapter, even though it may not match the quantum RG for quantum critical systems.

We now consider a method for explicitly constructing the tensor elements of the MERA network. We first introduce the following representation with a singlet and triplet on each of the two sites:

$$\sqrt{12}\,|\psi\rangle = 3\,|s\rangle_{23} \otimes |s\rangle_{41} + |t_0\rangle_{23} \otimes |t_0\rangle_{41} - |t_+\rangle_{23} \otimes |t_-\rangle_{41} - |t_-\rangle_{23} \otimes |t_+\rangle_{41}\,, \tag{5.28}$$

where $|s\rangle = (|\uparrow\downarrow\rangle - |\downarrow\uparrow\rangle)/\sqrt{2}, |t_0\rangle = (|\uparrow\downarrow\rangle - |\downarrow\uparrow\rangle)/\sqrt{2}, |t_+\rangle = |\uparrow\uparrow\rangle$, and $|t_-\rangle = |\downarrow\downarrow\rangle$. The purpose of this representation is that it considers the nonlocal bases and connects them to disentangling tensors (see Fig. 5.12).

We now introduce the disentangling transformation $|s\rangle \to |00\rangle$, $|t_+\rangle \to |01\rangle$, $|t_-\rangle \to |10\rangle$, and $|t_0\rangle \to |11\rangle$. This is a unitary transformation that locally reduces the amount of entanglement. This change is quite powerful for understanding the properties of the MERA network. Then, we have

$$\begin{aligned}\sqrt{12}\,|\psi\rangle &= 3\,|00\rangle_{23}\otimes|00\rangle_{41} + |11\rangle_{23}\otimes|11\rangle_{41} - |10\rangle_{23}\otimes|01\rangle_{41} - |01\rangle_{23}\otimes|10\rangle_{41} \\ &= 3\,|0000\rangle_{1234} + |1111\rangle_{1234} - |1100\rangle_{1234} - |0011\rangle_{1234} \\ &= \left(|00\rangle_{12}\ |11\rangle_{12}\right)\begin{pmatrix}3 & -1\\ -1 & 1\end{pmatrix}\begin{pmatrix}|00\rangle_{34}\\ |11\rangle_{34}\end{pmatrix}. \end{aligned} \tag{5.29}$$

The vector $|aa\rangle$ $(a = 0, 1)$ can be simply represented as $|a\rangle$, which compresses the information. The MERA representation of the ground state corresponds to the decomposition of the coefficient $\psi^{s_1s_2s_3s_4}$ by a set of functional tensors:

$$\psi^{s_1s_2s_3s_4} = \sum_{i_1,i_2}\sum_{j_1,j_2,j_3,j_4} T^{i_1i_2}W_{i_1}^{j_1j_2}W_{i_2}^{j_3j_4}U_{j_2j_3}^{s_2s_3}U_{j_4j_1}^{s_4s_1}. \tag{5.30}$$

Here we assume a spatially uniform network. Thus, the top tensor is defined by

$$T = \frac{1}{\sqrt{12}}\begin{pmatrix}3 & -1\\ -1 & 1\end{pmatrix}, \tag{5.31}$$

and the isometry tensor is defined by

$$W_i^{jj'} = \delta_{ij}\delta_{jj'}\ ,\ i = 0, 1. \tag{5.32}$$

The disentangling tensor is defined by

$$\begin{pmatrix}|00\rangle\\ |01\rangle\\ |10\rangle\\ |11\rangle\end{pmatrix} = \begin{pmatrix}|s\rangle\\ |t_+\rangle\\ |t_-\rangle\\ |t_0\rangle\end{pmatrix} = U\begin{pmatrix}|\uparrow\uparrow\rangle\\ |\uparrow\downarrow\rangle\\ |\downarrow\uparrow\rangle\\ |\downarrow\downarrow\rangle\end{pmatrix}\ ,\ U = \begin{pmatrix}0 & 1/\sqrt{2} & -1/\sqrt{2} & 0\\ 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 1/\sqrt{2} & 1/\sqrt{2} & 0\end{pmatrix}. \tag{5.33}$$

The matrix elements of U originate from the combination of $|s\rangle$ and $|t_0\rangle$, which resembles the Haal wavelet transformation. Thus, we conclude that the quantum RG can also be regarded as an extension of the scale control techniques that have been developed in classical systems.

5.5.4 RG Flow

The effective two-site Hamiltonian after the quantum RG flow using one layer of the disentangling and isometric transformations is obtained by taking the partial expectation value:

$$H^{\text{eff}}_{(i'_1 i'_2)(i_1 i_2)} = \sum_{s_1,s_2,s_3,s_4} \sum_{s'_1,s'_2,s'_3,s'_4} \left\langle s'_1 s'_2 s'_3 s'_4 \left| H \right| s_1 s_2 s_3 s_4 \right\rangle U^{s'_2 s'_3}_{i'_1 i'_2} U^{s'_4 s'_1}_{i'_2 i'_1} U^{s_2 s_3}_{i_1 i_2} U^{s_4 s_1}_{i_2 i_1} . \tag{5.34}$$

The matrix representation of the effective Hamiltonian and vector representation of the top tensor are given by

$$H^{\text{eff}} = \begin{pmatrix} -3/2 & 1/2 & 1/2 & -1/2 \\ 1/2 & 0 & 0 & 1/2 \\ 1/2 & 0 & 0 & 1/2 \\ -1/2 & 1/2 & 1/2 & 1/2 \end{pmatrix} , \; |T\rangle = \frac{1}{\sqrt{12}} \begin{pmatrix} 3 \\ -1 \\ -1 \\ 1 \end{pmatrix} , \tag{5.35}$$

respectively, the basis set is $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$. The ground state energy is given by

$$E = \left\langle T \left| H^{\text{eff}} \right| T \right\rangle = \langle \psi \left| H \right| \psi \rangle = -2J. \tag{5.36}$$

Here, the eigenvalues of H^{eff} are $E = -2J, 0, 0, J$.

We now calculate the bipartite entanglement entropy after the RG, which is defined by the entanglement entropy between the two effective sites (isometry tensors). The reduced density matrix is defined by

$$\rho_R = tr_L \left|T\right\rangle \left\langle T\right| = \frac{5}{6} \left|0\right\rangle \left\langle 0\right| - \frac{1}{3} \left|0\right\rangle \left\langle 1\right| - \frac{1}{3} \left|1\right\rangle \left\langle 0\right| + \frac{1}{6} \left|1\right\rangle \left\langle 1\right| = \begin{pmatrix} 5/6 & -1/3 \\ -1/3 & 1/6 \end{pmatrix} . \tag{5.37}$$

Here, the eigenvalues of ρ_R are $\left(3 \pm 2\sqrt{2}\right)/6$. The entropy is then calculated as

$$S_R = -\text{Tr}_R \left(\rho_R \log \rho_R\right) = \log 6 - \frac{2\sqrt{2}}{3} \log \left(3 + 2\sqrt{2}\right) = 0.12984 \cdots . \tag{5.38}$$

Thus, we determine that $S_R < S_{12}$. Hence, the disentangling procedure actually reduces the entanglement entropy, as discussed in the original paper by Vidal [21].

5.5.5 Concluding Remarks

In this section, we presented the analytic properties of the MERA network to better understand the nature of the quantum RG. Although general cases with a larger N are still difficult, the present toy model largely demonstrates how the RG occurs in quantum cases. The practical use of a nonlocal basis to decompose the wave function is key to constructing the tensor network.

5.6 Summary

In this chapter, we presented two applications of tensor networks for tensor data processing and a discussion of the underlying mechanism of the success of a tensor network related to the compression of quantum information (MERA). Regarding further applications, parameter compression of neural networks by tensor networks is also interesting [10]; however, we could not introduce this here due to space limitations. As discussed in this chapter, research on tensor data processing using tensor networks is promising, and its future development is necessary to support next-generation mobility technologies.

References

1. S. Cheng, L. Wang, T. Xiang, P. Zhang, Tree tensor networks for generative modeling. Phys. Rev. B **99**(15), 155131 (2019). https://doi.org/10.1103/physrevb.99.155131
2. A. Cichocki, Tensor networks for big data analytics and large-scale optimization problems (2014). arXiv:1407.3124
3. J. Eisert, M. Cramer, M.B. Plenio, Colloquium: Area laws for the entanglement entropy. Rev. Mod. Phys. **82**, 277–306 (2010). https://doi.org/10.1103/RevModPhys.82.277
4. Z.-C. Gu, M. Levin, X.-G. Wen, Tensor-entanglement renormalization group approach as a unified method for symmetry breaking and topological phase transitions. Phys. Rev. B **78**, 205116 (2008). https://doi.org/10.1103/PhysRevB.78.205116
5. Z.-Y. Han, J. Wang, H. Fan, L. Wang, P. Zhang, Unsupervised generative modeling using matrix product states. Phys. Rev. X **8**(3), 031012 (2018). https://doi.org/10.1103/physrevx.8.031012
6. K. Harada, T. Okubo, N. Kawashima, Network optimization of tree generative models (in preparation)
7. T.G. Kolda, B.W. Bader, Tensor decompositions and applications. SIAM Rev. **51**(3), 455–500 (2009). https://doi.org/10.1137/07070111X
8. C.H. Lee, X.-L. Qi, Exact holographic mapping in free fermion systems. Phys. Rev. B **93**, 035112 (2016). https://doi.org/10.1103/PhysRevB.93.035112
9. T. Nishino, Y. Hieida, K. Okunishi, N. Maeshima, Y. Akutsu, A. Gendiar, Two-dimensional tensor product variational formulation. Prog. Theor. Phys. **105**(3), 409–417 (2001). https://doi.org/10.1143/ptp.105.409
10. A. Novikov, D. Podoprikhin, A. Osokin, D.P. Vetrov, Tensorizing neural networks. Adv. Neural Inf. Proc. Syst. **28**, 442–450 (2015)
11. T. Okubo, N. Kawashima (in preparation)
12. K. Okunishi, T. Nishino, H. Ueda, Developments in the tensor network – from statistical mechanics to quantum entanglement. J. Phys. Soc. Jpn. **91**(6), 062001 (2022). https://doi.org/10.7566/JPSJ.91.062001
13. R. Orús, A practical introduction to tensor networks: matrix product states and projected entangled pair states. Ann. Phys. **349**, 117–158 (2014). https://doi.org/10.1016/j.aop.2014.06.013
14. I.V. Oseledets, Tensor-train decomposition. SIAM J. Sci. Comput. **33**(5), 2295–2317 (2011). https://doi.org/10.1137/090752286
15. X.-L. Qi, Exact holographic mapping and emergent space-time geometry (2013). arXiv:1309.6282
16. R. Salakhutdinov, Learning deep generative models. Annu. Rev. Stat. Appl. **2**(1), 361–385 (2015). https://doi.org/10.1146/annurev-statistics-010814-020120
17. Y.Y. Shi, L.M. Duan, G. Vidal, Classical simulation of quantum many-body systems with a tree tensor network. Phys. Rev. A **74**(2), 022320 (2006). https://doi.org/10.1103/physreva.74.022320

18. E. Stoudenmire, D.J. Schwab, Supervised learning with tensor networks. Adv. Neural Inf. Proc. Syst. **29**, 4799–4807 (2016)
19. B. Swingle, Entanglement renormalization and holography. Phys. Rev. D **86**, 065007 (2012). https://doi.org/10.1103/PhysRevD.86.065007
20. F. Verstraete, J.I. Cirac, Renormalization algorithms for quantum-many body systems in two and higher dimensions (2004). arXiv:cond-mat/0407066
21. G. Vidal, Entanglement renormalization. Phys. Rev. Lett. **99**, 220405 (2007). https://doi.org/10.1103/PhysRevLett.99.220405
22. S.R. White, Density matrix formulation for quantum renormalization groups. Phys. Rev. Lett. **69**, 2863–2866 (1992). https://doi.org/10.1103/PhysRevLett.69.2863
23. S.R. White, Density-matrix algorithms for quantum renormalization groups. Phys. Rev. B **48**, 10345–10356 (1993). https://doi.org/10.1103/PhysRevB.48.10345
24. Q. Zhao, G. Zhou, S. Xie, L. Zhang, A. Cichocki, Tensor ring decomposition (2016). arXiv:1606.05535

Chapter 6
Machine Learning Approach to Mobility Analyses

Kazushi Ikeda and Takatomi Kubo

6.1 Introduction

Sociality is an essence of life in many species such as ants, bees, monkeys, and humans. Sociality is characterized by the interactions among the individuals in a group and/or those among groups, which appear in their behavior such as movement. The movement of agents has easily been recorded thanks to the IoT technologies and produces a huge amount of data.

Recent technologies for mobility are so innovative that they have been changing even sports scenes [1]. However, the tools for science are not sufficiently developed since any sensors are not equipped on agents in scientific experiments or wildlife observations. In addition, the interactions among agents are not objectively analyzed so far due to the lack of tools.

In short, therefore, we need the three steps below for the mobility analyses. The first step is to track and identify agents therein. The second step is to extract interactions from the tracking data. The last step is to analyze graphs since the interactions are represented as a graph, where an agent and an interaction correspond to a node and an edge, respectively.

This chapter introduces some tools recently developed for the three steps above. As an example of the first step, we introduce a deep learning-based multi-animal tracking system. The system is named Deep MAnTra and is applicable to non-human primates studies [8], since they experience stress and show less movement when they are relocated and socially isolated [7, 12]. As an example of the second

K. Ikeda (✉) · T. Kubo
Nara Institute of Science and Technology, 8916-5 Takayama-cho, Ikoma, Nara 630-0192, Japan
e-mail: kazushi@is.naist.jp

T. Kubo
e-mail: takatomi-k@is.naist.jp

K. Ikeda et al. (eds.), *Advanced Mathematical Science for Mobility Society*,
https://doi.org/10.1007/978-981-99-9772-5_6

step, we introduce a method for horse herding [2]. An example of the last step is a technique for graph neural networks called MLAP (multi-level attention pooling) [5]. These tools help analyze mobility data given as videos.

6.2 Deep Learning-Based Multi-animal Tracking

The first step of the mobility analyses is to track and identify agents in videos. In this section, we introduce a deep learning-based multi-animal tracking system called the Deep MAnTra (Fig. 6.1) [8]. Multi-animal tracking in a wild environment is a considerably hard challenge because animals show agile motions, however, it is an active research problem in computer vision, since this kind of methods is not only applicable to animals but also able to cover the mobility analyses.

In Deep MAnTra, a monkey detector using You Only Look Once (YOLOv4) [9] is trained with carefully designed transfer learning. Using the resulting box detections from our monkey detection model, SuperGlue [10] and Murty's algorithm are used for re-identifying the monkey individuals across the succeeding frames.

As a result, the Japanese macaque detection model trained using a YOLOv4 architecture with spatial attention module, combined with the Mish activation function based on a 3-stage training curriculum yielded the best performance with a mean AP50 of 96.59%, a precision score of 93%, a recall of 96%, and a mean IOUAP@50 of 77.2%. Deep MAnTra can prove effective and reliable for animal behavior studies, since it achieved 91.35% MOTA (Multiple Object Tracking Accuracy) even on a heterogeneous dataset.

6.3 Horse Herding Analysis

The second step is to extract interactions from the tracking data. In this section, we introduce an example of interaction analyses, which proposed a mathematical model of feral horses [2].

The purpose of this model is to study the mechanism of group aggregation by herding, which is necessary for a stallion (the male leader horse of a group) to keep his harem of mares (female horses). Herding is also seen between a shepherd dog and a sheep, where the shepherd dog moves to aggregate a group of sheep and move the group in a certain direction with two crucial strategies [11]. While herding, different forces of interaction come into play, reflecting the dynamics of the movements of the sheep and the strategies of the dog. The herding of sheep by a shepherd dog is explained as a social force model [3] comprising self-propelled particles with a constant speed. In the case of mares, they do not move with a constant speed. Thus, we proposed a social force model where the motion is a linear combination of various forces such as repulsion from the stallion and attraction to the center of mass.

The forces employed in our social force model are listed below (Fig. 6.2).

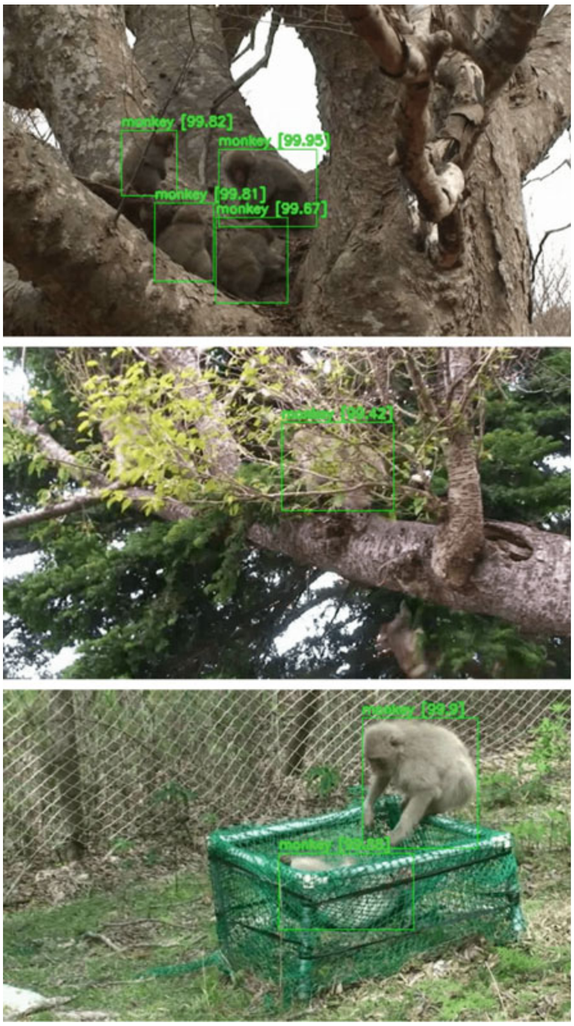


Fig. 6.1 Image examples of multiple monkey detection with Deep MAnTra. Each green box indicates the region showing a monkey. Our approach successfully detected monkeys in these examples even if they are occluded

Inertia:	A mare tends to move in the same direction as the previous one.
Repulsion from the stallion:	When a mare is within a certain distance from the stallion, she experiences a force of repulsion directed away from the stallion.
Short-range repulsion:	A mare experiences a force of repulsion from other mares when they are within the repulsion zone, which means that a mare tends to retain an exclusive zone around her.

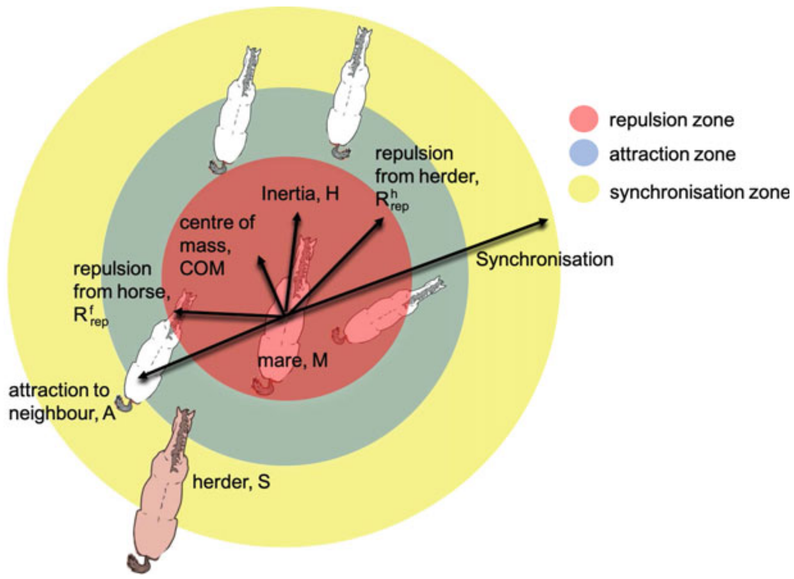


Fig. 6.2 Schematic diagram of the zones and forces of interaction experienced by a mare while being herded by a stallion [2]

Medium-range attraction:	A mare experiences a force of attraction from other mares when they are within the attraction zone, which means that a mare likes to match its direction to her specific zone.
Synchronization attraction:	A mare matches her direction with the nearest moving mare within a set region, which means that a moving mare may be escaping from danger.
Attraction to the center of mass:	Mares experience a force of attraction toward the group's center of mass, since divergent positions increase the risk of predation.

The driving force of a mare is a weighted sum of these components. We assumed that each mare has specific weights and estimated them by minimizing the squared error between the trajectories of the model and the measurements subject to the condition that the coefficients are non-negative, which results in a non-negative factorization problem.

The estimated weights and the reproduced trajectories are in Table 6.1 and in Fig. 6.3, respectively. We can see two points from the weights. One is that the weights c_4 and c_5 take small values compared to the others, which means that these components are ignorable in our model. This is an advantage of using the non-negative optimization. The other is that the weights of the mares are widely distributed, except for Mares 1 and 2. This variation may indicate forms of relationships among individuals within the harem. In fact, Mares 1 and 2 are sisters.

Table 6.1 Weights of the forces obtained

Mare identity	Inertia, c_1	Repulsion from the stallion, c_2	Short-range repulsion, c_3	Medium-range attraction, c_4	Synchronization attraction, c_5	Attraction to COM, c_6
1	0.600	0.009	0	0	0.134	0.051
2	0.499	0.183	0.033	0	0.083	0.036
3	0.353	0.399	0.204	0.024	0.015	0.592
4	0.595	0.359	0.230	0	0.125	0.423
5	0.912	0.399	0.037	0	0	0.287
6	0.677	0.272	0	0.027	0	0.298

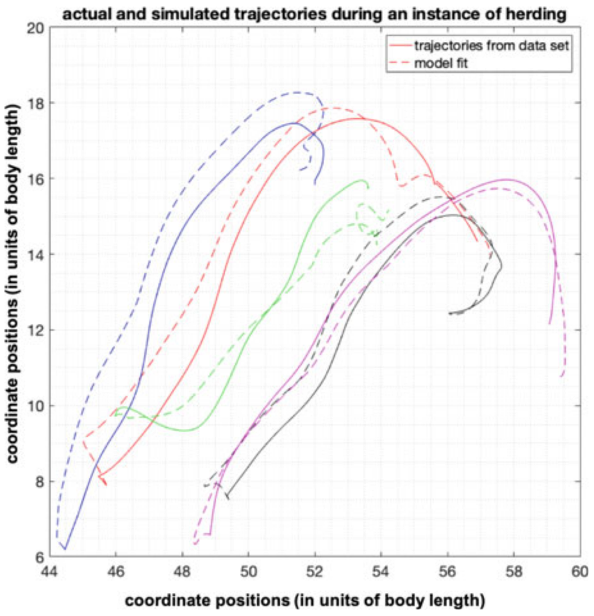


Fig. 6.3 Comparison of actual mare trajectories with the best-fit model

In this section, we introduced a mathematical model study for herding in mares as an example of interaction analyses. The social force model used therein is general, easy to use, and explainable for interaction analyses. In addition, the weights represent a personality of the agent and then are useful for further analyses.

6.4 Multi-level Attention Pooling for Graph Neural Networks

As an example of machine learning techniques for graphs, we introduce deep graph neural networks called MLAP (multi-level attention pooling) [5].

Graph-structured data are found in many fields since a wide variety of natural and artificial objects can be expressed with graphs, such as molecular structural formula, biochemical reaction pathways, brain connection networks, social networks, and abstract syntax trees of computer programs as well as mobility analysis. Because of this ubiquity, machine learning methods on graphs have been actively studied. Thanks to rich information underlying the structure, graph machine learning techniques have shown remarkable performances in various tasks.

In contrast to classical graph machine learning methods using hand-crafted features, recent years have witnessed a surge in graph representation learning (GRL). Recently, graph neural networks (GNNs) have rapidly emerged as a new framework for GRL.

Itoh et al. have proposed a method to learn graph representation for graph-level prediction tasks, such as molecular property classification, by using multiple representations in different localities [5]. For this representation learning, they proposed a multi-level attention pooling (MLAP) architecture that introduces an attention pooling layer [6] for each message passing step to compute layer-wise graph representations. Then, the proposed architecture aggregates them to compute the final graph representation. Thus, the MLAP architecture can focus on different nodes (or different subgraphs) in each layer with different levels of information localities, which leads to better modeling of both local structural information and global structural information (Fig. 6.4).

The MLAP architecture is shown to improve the graph classification performance compared to the baseline architectures [5]. In addition, analyses of the layer-wise graph representations suggest that aggregating information from multiple levels of

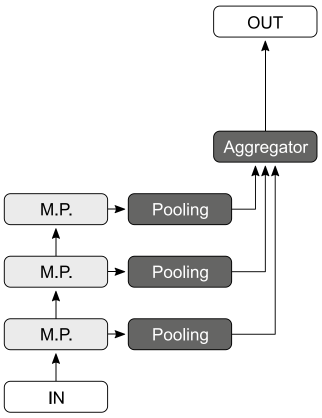


Fig. 6.4 Architecture of a GNN with MLAP. There is a dedicated pooling layer for each message passing layer to compute layer-wise graph representation. The aggregator computes the final graph representation from the layer-wise graph representations. M.P.: message passing

localities indeed has the potential to improve the discriminability of learned graph representations.

The MLAP architecture is applicable not only to classification task, but also to sequence generation task (Graph2Seq) [4]. Further studies will widen the applicability more in the future.

6.5 Conclusions

This chapter introduced some tools for three steps for mobility analyses. The first is a multi-animal tracking system based on deep learning techniques called Deep MAnTra [8]. The second is a social force model that extracts interactions from the tracking data as well as characterizes the individuals, applied to horse herding [2]. The last is a graph neural network based on a representation learning called MLAP [5]. These techniques will help us analyze mobility data given as videos.

References

1. FIFA: Football technologies & Innovations at the FIFA World Cup Qatar (2022). https://www.fifa.com/technical/football-technology/
2. C.K. Go, M. Ringhofer, B. Lao, T. Kubo, S. Yamamoto, K. Ikeda, A mathematical model of herding in horse-harem group. J. Ethol. **38**(3), 343–353 (2020)
3. D. Helbing, P. Molnár, Social force model for pedestrian dynamics. Phys. Rev. E **51**(5), 4282–4286 (1995)
4. T.D. Itoh, T. Kubo, K. Ikeda, Compositionality-aware Graph2Seq learning (2022). arXiv:2201.12178
5. T.D. Itoh, T. Kubo, K. Ikeda, Multi-level attention pooling for graph neural networks: unifying graph representations with multiple localities. Neural Netw. **145**, 356–373 (2022)
6. Y. Li, D. Tarlow, M. Brockschmidt, R. Zemel, Gated graph sequence neural networks, in *Proceedings of the 4th International Conference on Learning Representations* (2016)
7. J.S. Meyer, A.F. Hamel, Models of stress in nonhuman primates and their relevance for human psychopathology and endocrine dysfunction. ILAR J. **55**, 347–360 (2014)
8. R.R. Pineda, T. Kubo, M. Shimada, K. Ikeda, Deep MAnTra: deep learning-based multi-animal tracking for Japanese macaques. Artif. Life Robot. **28**(1), 127–138 (2023)
9. J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 779–788
10. P.-E. Sarlin, D. DeTone, T. Malisiewicz, A. Rabinovich, Superglue: learning feature matching with graph neural networks, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 4938–4947
11. D. Strömbom, R.P. Mann, A.M. Wilson, S. Hailes, A.J. Morton, D.J.T. Sumpter, A.J. King, Solving the shepherding problem: heuristics for herding autonomous, interacting agents. J. R. Soc. Interface **11**(20140719) (2014)
12. S.L. Willard, C.A. Shively, Modeling depression in adult female cynomolgus monkeys (Macaca fascicularis). Am. J. Primatol. **74**, 528–542 (2012)

Chapter 7
Graph Optimization Problems and Algorithms for DAG-Type Blockchains

Jun Kawahara

7.1 Introduction

A large amount of data must be processed when utilizing mobility technologies. For example, in car sharing, a large amount of data are generated, including information on rented vehicles, such as who is riding in which vehicle, when and where the vehicle travels, and the history of money received and paid. A database that stores such data must be fault-tolerant. A centralized database that manages all data in a single location results in massive confusion when a failure of that database occurs. Therefore, data should be managed in a distributed manner as much as possible; one way to do this is by using a distributed database.

Although many distributed database technologies exist, blockchain technology has been attracting attention in recent years [9]. Blockchain is primarily used to realize virtual currencies without a centralized administrator. However, the simple line-type blockchain used in Bitcoin and other applications is limited by the amount of data that can be stored; it cannot withstand the large amount of data that is typical for mobility applications.

A directed acyclic graph (DAG)-type blockchain has been proposed to increase the amount of data stored in the blockchain and improve scalability [8] (see Fig. 7.1). In a line-type blockchain, transactions of blocks of a chain other than the longest blockchain are discarded, which is inefficient. In a DAG-type blockchain, fewer transactions are discarded, which is expected to be more efficient.

In blockchain technology, dealing with malicious participants is a challenge. As a blockchain is realized using P2P network technology, all participants in the network play the same role. Any participant can create and add blocks to the blockchain that are favorable to them. In a line-type blockchain, a mechanism called "proof-of-work" [9] makes adding malicious blocks difficult; however, in a DAG-type blockchain, mali-

J. Kawahara (✉)
Kyoto University, Yoshida-honmachi, Sakyo-ku, Kyoto 606-8501, Japan
e-mail: jkawahara@i.kyoto-u.ac.jp

K. Ikeda et al. (eds.), *Advanced Mathematical Science for Mobility Society*,
https://doi.org/10.1007/978-981-99-9772-5_7

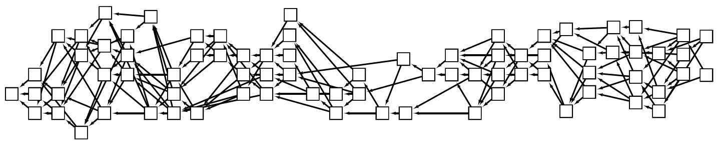


Fig. 7.1 Example of a DAG-type blockchain

cious blocks must be detected differently. One method is to identify trusted blocks by solving some type of independent set problem on a DAG. However, the independent set problem is NP-hard and is difficult to solve quickly.

In this chapter, we consider a DAG-type blockchain as a graph structure and analyze it based on graph algorithm theory. Many participants in a P2P network are expected to connect their blocks to those near the tail of the DAG-type blockchain. Therefore, a DAG-type blockchain is expected to have a structure that is similar to that of a directed path. In undirected graphs, a characteristic called the pathwidth [11] has been studied, which indicates how much a graph is close to a path. Various algorithms can deal with undirected graphs with small pathwidths efficiently. Several similar metrics exist for directed graphs [3, 6, 10], but they cannot be applied to DAGs. This chapter introduces the concept of the DAG-pathwidth introduced in the paper [7], in which we designed an algorithm to efficiently solve some independent set problems on DAGs by performing DAG-path decomposition. Note that finding a DAG-path decomposition with a minimum DAG-pathwidth is NP-hard [7].

7.1.1 Related Work

The concept of the pathwidth was first proposed by Neil et al. [11]. Often, the treewidth is used to measure the width of undirected graphs (proposed by Neil et al. [12] and Rudolf et al. [5]), which represents the closeness of a graph to a tree. Many algorithms using treewidth have been proposed [2]. Calculating both pathwidths and treewidths is NP-complete [1].

Researchers have proposed various width measures for directed graphs, such as the directed pathwidth [10], directed treewidth [6], and DAG-width [3]. The reason for introducing a new graph parameter is that these existing widths are unsuitable for DAGs. This is the main difference between existing widths and our proposed width. Some studies have solved problems for a directed graph with a small width, for example, deciding the reachability of DAGs [4]. Sompolinsky and Zohar [13] formulated a problem for the DAG-type blockchain; they proposed a DAG-type protocol called PHANTOM. This is called a discord k-independent set problem.

7.2 Preliminaries

In this section, we introduce some definitions and notation. A directed graph is a pair $D = (N, A)$, where N is a set of elements, called *nodes*, and $A \subseteq N \times N$ is a pair of nodes, called *arcs*. For an arc $(u, v) \in A$, u is called the *tail* and v is called the *head*. For two nodes $v_1, v_a \in N$, if there are nodes $v_2, \dots, v_{a-1}$ such that $(v_i, v_{i+1}) \in A$ for $i = 1, \dots, a-1$, then $(\{v_1, \dots, v_a\}, \{(v_i, v_{i+1}) \mid i = 1, \dots, a-1\})$ is a *directed path*, and we say that v_a is reachable from v_1. We also write $v_1 \rightsquigarrow_D v_a$. If $(v_a, v_1) \in A$ additionally holds, $(\{v_1, \dots, v_a\}, \{(v_i, v_{i+1}) \mid i = 1, \dots, a-1\} \cup \{(v_a, v_1)\})$ is a *directed cycle*. A *directed acyclic graph* (*DAG*) is a directed graph that has no directed cycle. For an arc subset $A' \subseteq A$, if every two nodes u, v in A' are reachable to each other (i.e., $u \rightsquigarrow_D v$ and $v \rightsquigarrow_D u$), A' is called a *strongly connected component*. Every strongly connected component of a DAG consists of exactly one node (otherwise, the DAG has a directed cycle).

For a node subset $N' \subseteq N$, the *induced subgraph* of N' is a graph whose node set is N' and whose arc set consists of the arcs such that both the head and tail are in N'. We denote by $D[N']$ the induced subgraph of N'. That is, $D[N'] = (N', \{(u, v) \in A \mid u \in N', v \in N'\})$.

7.3 Blockchain

This section provides the minimum necessary modeling of the blockchain. This is a greatly simplified version of the blockchain used in Bitcoin and other virtual currencies. First, we describe a blockchain whose shape looks like a chain, used in Bitcoin and other currencies, followed by a blockchain whose shape looks like a DAG. The former is called a chain-type blockchain (called line-type in the introduction) and the latter a DAG-type blockchain.

7.3.1 Chain-Type Blockchain

A blockchain operates as a distributed database that holds data generated by multiple participants in a consistent manner. There are multiple participants on the network that generate data such as money transfer transactions. The generated data is called a *piece*. A piece can be considered as a string of arbitrary length on $\{0, 1\}$, but we will not be concerned with the contents of the piece here. There are several *miners* on the network, apart from the participants. A participant who generates the data may also play the role of a minor. Participants broadcast the pieces they wish to record in the blockchain on the network, i.e., they send data to the miners on the network. When a minor receives a piece from a participant, the minor temporarily stores it in the minor's own memory, which is called a *pool*.

A minor attempts to create a block. A block is defined by a triple (h, P, c). The first element h is a natural number as described below. P is a set of pieces. The minor takes pieces from the pool when creating the block and stores them in the block as P. The third element c, called *nonce*, is a natural number at least 0 and less than U_{N}, whose value must be appropriately determined by the block creator in the manner described below, where U_{N} is a fixed value determined by the protocol, e.g., $U_{\mathrm{N}} = 2^{32}$ in the Bitcoin blockchain.

Before we describe in detail how to create a block, we introduce the notion that for two blocks $B_1 = (h_1, P_1, c_1)$ and $B_2 = (h_2, P_2, c_2)$, B_2 *refers to* B_1. Consider the (cryptographic) hash function $h^*(s) : \{0, 1\}^* \longrightarrow \{0, 1, \ldots, U_{\mathrm{H}} - 1\}$ whose domain is a string on $\{0, 1\}$ and whose value range is any natural number at least 0 and less than U_{H}, where $\{0, 1\}^*$ is the set of all strings on $\{0, 1\}$ and U_{H} is a fixed value determined by the protocol, e.g., $U_{\mathrm{H}} = 2^{256}$ in the Bitcoin blockchain. The hash function $h^*(s)$ is a function defined by the protocol. For example, in the Bitcoin blockchain, a function based on SHA-256 is used. Consider obtaining the hash function value of the block's data h_1, P_1, c_1. P_1 is a set of strings (pieces), and each element is concatenated using delimiters. The values h_1 and c_1 are natural numbers and are properly encoded in the string. Finally, h_1, P_1, and c_1 are concatenated using delimiters. We omit the specific encoding method. By making the string for the input to the hash function h^*, a natural number at least 0 and less than U_{H} is obtained as output. The output value is written as $h^*(B_1)$ or $h^*((h_1, P_1, c_1))$ with symbol abuse. We say that $h_2 = h^*((h_1, P_1, c_1))$ and further that B_2 *refers to* B_1 if $h_2 < T$ is satisfied for T that is defined later. For a block $B_i = (h_i, P_i, c_i)$, $i = 1, 2, \ldots$, a *chain* in a chain-type blockchain is a sequence of blocks $B_1, B_2, \ldots$ such that B_{i+1} refers to B_i. We assume that $h_1 = 0$ and $c_1 = 0$. We call B_1 the *genesis block*.

A minor creates a block and sends it by broadcast to another minor. A conscientious minor will make every effort to ensure that the block reaches the entire minor as much as possible. However, the arrival of the block may be delayed due to network congestion or other reasons. The set of blocks recognized by a minor m (i.e., generated by itself or received from other minors) is denoted by $\mathcal{C}_m$. Blocks are data, not objects, and data are copied, not moved. That is, when a minor m sends a block B to a minor m', B is contained in both $\mathcal{C}_m$ and $\mathcal{C}_{m'}$.

Block creation is performed as follows. A minor m obtains the longest chain from $\mathcal{C}_m$. That is, we consider the block sequence $B_1, \ldots, B_\ell \in \mathcal{C}_m$ with the maximum length ℓ such that B_{i+1} refers to B_i for $i = 1, 2, \ldots, \ell - 1$ and B_1 is the genesis block. The minor m attempts to create a block $B_{\ell+1} = (h_{\ell+1}, P_{\ell+1}, c_{\ell+1})$. Let $h_{\ell+1} = h^*(B_\ell)$. The set $P_{\ell+1}$ is created by taking several pieces from the pool of m. The value $c_{\ell+1}$ must be determined so that $h^*((h_{\ell+1}, P_{\ell+1}, c_{\ell+1})) < T$. Here, it is not computationally easy to determine $c_{\ell+1}$ so that the value of $h^*((h_{\ell+1}, P_{\ell+1}, c_{\ell+1}))$ is less than T. (The hash function h^* must be designed so.) We determine the value of $c_{\ell+1}$ by repeatedly and randomly choosing $c_{\ell+1}$ and calculating $h^*((h_{\ell+1}, P_{\ell+1}, c_{\ell+1}))$ until it is less than T. If a minor succeeds in finding such $c_{\ell+1}$, i.e., generating $B_{\ell+1}$ that refers to B_ℓ, the minor m removes all pieces of $P_{\ell+1}$ from the pool, adds $B_{\ell+1}$ to $\mathcal{C}_m$, and broadcasts $B_{\ell+1}$. Another minor m' that receives $B_{\ell+1}$ adds $B_{\ell+1}$ to $\mathcal{C}_{m'}$. The minors m and m' attempt to create the next block in a similar process.

Miners are given an incentive to generate blocks. In the Bitcoin blockchain, bitcoins are awarded to a minor who creates a block B (that information is written to B). Thus, miners compete to create a block. The process of generating blocks is called *mining*. The value of T mentioned above is called the *target*. In Bitcoin, T is adjusted so that the average time interval for one of all miners to successfully create a block (to find a nonce that satisfies the condition) is ten minutes. The target T at the creation of the block $B_{\ell+1}$ following the chain $B_1, \dots, B_\ell$ depends on the information contained in $B_1, \dots, B_\ell$.

If many miners are conscientious, many miners are expected to have the same chain $B_1, \dots, B_\ell$. However, blocks near the end of the chain may not have arrived due to network congestion. There is an incentive to create blocks because there is an incentive to maintain the blockchain network. Therefore, many miners can be expected to be conscientious. The data stored in the blockchain is interpreted as $P_1, \dots, P_\ell$ (this is not a mathematical definition).

When creating a new block $B_{\ell+1} = (h_{\ell+1}, P_{\ell+1}, c_{\ell+1})$, a minor must check whether each piece in $P_{\ell+1}$ is consistent with $P_1, \dots, P_\ell$, and the minor does not make an inconsistent piece belong to $P_{\ell+1}$. A block $B_{\ell+1}$ whose $P_{\ell+1}$ is inconsistent with $P_1, \dots, P_\ell$ will be ignored by the other miners. Therefore, the miner correctly tries to create $B_{\ell+1}$ in order to gain incentives. This mechanism ensures the consistency of data in the blockchain. Pieces in a block that are out of the longest chain are considered invalid. Note that $P_1, \dots, P_\ell$ will never be fixed, as longer chains may be created later. In practice, however, for a sufficiently large ℓ, overwriting all of $B_1, \dots, B_\ell$ would require the creation of a chain longer than ℓ and would require a large amount of computing power to compute the hash function. Miners with the ability to create chains longer than ℓ would have less incentive to overwrite all of $B_1, \dots, B_\ell$ because they would be incentivized to create chains behind B_ℓ according to the legitimate rules.

Two blocks may be generated simultaneously by different miners. Suppose that two blocks, $B_{\ell+1}$ and $B'_{\ell+1}$, are generated behind B_ℓ. Many miners receive both $B_{\ell+1}$ and $B'_{\ell+1}$, but arbitrarily choose one of them (usually the block that arrived first) and attempt to create the block following the one they chose. If another miner successfully creates and broadcasts a block $B_{\ell+2}$ behind $B_{\ell+1}$, for example, $B_1, \dots, B_\ell, B_{\ell+1}, B_{\ell+2}$ is longer than $B_1, \dots, B_\ell, B'_{\ell+1}$. Then, most miners try to create a block following $B_{\ell+2}$. Even if a temporary chain split (called a *fork*) occurs, this mechanism often converges the fork. (Of course, it is not guaranteed that forks will converge, and the continued simultaneous creation of blocks will create chaos in the chain.)

7.3.2 DAG-Type Blockchain

Chain-type blockchains have the disadvantage that blocks that are out of the longest chain are discarded. In order to increase the amount of data stored, DAG-type blockchains were considered [8]. A DAG-type blockchain is a type of blockchain in which the directed graph composed of block references is a DAG rather than a directed path. We allow a block to refer to multiple blocks. We extend the definition of a block to $B = (H, P, c)$, where P is a set of pieces, c is a nonce, and H is a sequence of integer values at least 0 and less than U_{H}. For blocks $B^1, \dots, B^a$, we say that a block B refers to blocks $B^1, \dots, B^a$ if $h^i = h^*(B^i)$, $i = 1, \dots, a$ and $H = (h^1, \dots, h^a)$.

For simplicity, let $B_{\mathrm{gen}} = (\emptyset, \emptyset, 0)$ denote the genesis block in a DAG-type blockchain, and assume that all miners hold B_{gen}. The concept of a chain in a chain-type blockchain corresponds to a connected DAG containing the genesis block in a DAG-type blockchain. For two blocks $B, B' \in \mathcal{C}_m$ recognized by a minor m, if there exist blocks $B_1, B_2, \dots, B_a$, where $B_1 = B$, $B_a = B'$, and B_{i+1} refers to B_i for $i = 1, \dots, a-1$, we say that B is *reachable* from B'. Strictly speaking, B is said to be reachable from B' if $B_i = (H_i, P_i, c_i)$ and for $i = 1, \dots, a-1$, the value $h^*(B_i)$ is contained in the sequence H_{i+1}. Consider the set of blocks in $\mathcal{C}_m$ consisting of all blocks reachable to B_{gen}. We write it as $\mathcal{B}(\mathcal{C}_m)$. We also write $D = (N, A)$ for the DAG composed of $\mathcal{B}(\mathcal{C}_m)$ (as a graph structure). The elements of N are called nodes, and N has a one-to-one correspondence with $\mathcal{B}(\mathcal{C}_m)$. For an arc $(u, v) \in A$, let B_u and B_v denote the blocks corresponding to u and v, respectively. Then, B_u refers to B_v. D has no directed cycle because when B_u refers to B_v, the time when B_v is generated is before the time when B_u is generated. From the way D is created, there is exactly one node whose outgoing degree is 0, which corresponds to B_{gen}. We write the node as v_{gen}.

7.3.3 PHANTOM Protocol

When a block B refers to a block B', the creator of the block B believes that the piece held in the block B' is correct (consistent with the other pieces). Similarly, when a block B is reachable to a block B'', the creator of the block B believes that the piece held in the block B'' is correct. Thus, receiving references from many blocks would increase the reliability of the block. If many conscientious miners properly select and refer to blocks behind the DAG, the DAG will become closer and closer to the chain, and blocks closer to the beginning of the chain will accumulate trust. This makes the entire DAG-type blockchain more resistant to tampering and the system more stable.

However, accepting every block in the DAG as a legitimate block causes the following problem. A malicious minor m_{e} creates a block B containing a piece beneficial to him/her, but does not broadcast B and keeps it secret from others. He/she generates a new piece beneficial to himself/herself, creates a block, and makes it refer to B. In this way, m_{e} generates a block containing a piece beneficial to himself/herself one after another as a successor to B. When enough blocks have been created, m_{e} broadcasts B and the subsequent blocks simultaneously. In the mechanism that recognizes all blocks in the DAG as legitimate blocks, blocks created in this way are recognized as legitimate blocks. As another example, in an extreme case, B_{gen} could be referred to from all blocks created. Blocks that have not earned the trust of other miners are treated as legitimate blocks without restriction.

The PHANTOM protocol [13] of the DAG-type blockchain solves this problem in the following way. Before this description, we define some terms. In the following, consider a DAG $D = (N, A)$ corresponding to $\mathcal{B}(\mathcal{C}_m)$ of some minor m. For a node $v \in N$ of $D = (N, A)$, let $\mathsf{past}(v, D)$ be the set of nodes reachable from v on D. That is, $\mathsf{past}(v, D) = \{u \in N \mid D = (N, A), v \rightsquigarrow_D u\}$. Also, let $\mathsf{future}(v, D)$ be the set of nodes reachable to v on D. That is $\mathsf{future}(v, D) = \{u \in N \mid D = (N, A), u \rightsquigarrow_D v\}$. Let $\mathsf{anticone}(v, D)$ be the set of nodes in N that are contained in neither $\mathsf{past}(v, D)$ nor $\mathsf{future}(v, D)$. That is, $\mathsf{anticone}(v, D) = N \setminus \mathsf{past}(v, D) \setminus \mathsf{future}(v, D)$.

The PHANTOM protocol limits the number of blocks that have not earned the trust from other miners. Fix the natural number parameter k. For any subset $N' \subseteq N$ of nodes, N' is called a *discord k-independent set* if for any node $v \in N'$, $\mathsf{discord}(v, N', D) := |N' \cap \mathsf{anticone}(v, D)| \le k$. That is, for any node v in N', we restrict the number of nodes in N' that cannot be reached from v and cannot reach v to at most k. We consider the block corresponding to each node of N' as a valid block (for m), and consider the pieces contained in a valid block as valid. Blocks not contained in N' are considered invalid blocks, and pieces contained in invalid blocks are ignored. It is expected that this mechanism will prevent miners from creating blocks that are not referred to as blocks generated by other miners.

The PHANTOM protocol regards the discord k-independent set of D with the maximum number of nodes as the set of legitimate blocks. We need an efficient algorithm that computes the maximum discord k-independent set of D, which is discussed in Sect. 7.5.

7.4 Pathwidth

7.4.1 Pathwidth and Directed Pathwidth

Many fast methods exist for solving optimization problems. Although integer programming is a promising method, the problem we wish to solve must be solved by all minors. Therefore, fast commercial solvers for integer programming are not available. Since the problem we want to solve this time needs to be solved rigorously, approximation algorithms, heuristics, and genetic algorithms are not suitable. Here, we use parameterization algorithm techniques. Well-known techniques include the notions of pathwidth and treewidth for undirected graphs, as described in the introduction section. These are natural numbers that represent the closeness of a graph to a path or a tree. The smaller the value is, the closer the graph is to the path or tree. When pathwidth and treewidth are small, it is possible to decompose the graph into smaller pieces that do not interfere with each other, and faster algorithm design based on decomposition is expected. Since a newly generated block refers to blocks at the tail of the DAG, the shape of the DAG is expected to be close to a straight line. (Since the shape is not expected to be close to a tree, treewidth is not used.) The concept of directed pathwidth exists for directed graphs [10], but as will be explained later, it is not applicable to DAGs. We will introduce a parameter that can be applied to DAGs.

We describe path decomposition and pathwidth of an undirected graph. The path decomposition of an undirected graph $G = (V, E)$ is a sequence $(X_1, X_2, \ldots)$ of subsets of V satisfying the following conditions, where a subset X_i of V is called a *bag*:

(P1) $\bigcup_{i=1,2,\ldots} X_i = V$.
(P2) For every edge $(u, v) \in E$, there is a bag X_i such that $u, v \in X_i$.
(P3) For all i, j, k with $i < j < k$, $X_i \cap X_k \subseteq X_j$ holds.

(P1) is the condition that any element of V is contained in at least one of the bags $X_1, X_2, \ldots$. (P2) is the condition that there exists a bag containing both endpoints of any edge in E. (P3) means that for any node $v \in V$, the subscripts of the bags containing v are continuous as $X_i, X_{i+1}, \ldots, X_{k-1}, X_k$.

The pathwidth for a path decomposition $(X_1, X_2, \ldots)$ of G is defined by $\max_i\{|X_i| - 1\}$. Among the path decompositions of G, consider the path decomposition with the smallest pathwidth, and define the pathwidth of G as the pathwidth of the path decomposition. The -1 appearing in the definition of pathwidth is for adjusting the pathwidth of a path to be 1.

An important property of path decomposition is the following property. For any $i = 2, 3, \ldots$, consider the following. Let X_{L} be the set of nodes in the bag before X_i that are not included in X_i, and let X_{R} be the set of nodes in the bag after X_i that are not included in X_i. That is, $X_{\mathrm{L}} = \bigcup_{j<i} X_j \setminus X_i$ and $X_{\mathrm{R}} = \bigcup_{j>i} X_j \setminus X_i$. Given any nodes $x \in X_{\mathrm{L}}$ and $y \in X_{\mathrm{R}}$, the edge (x, y) is not contained in E. That is, X_{L} and X_{R} are separated by X_i. This is a property derived from (P1), (P2), and (P3).

Based on path decomposition, various problems can be solved efficiently. As an example, consider solving the maximum independent set problem. An independent set $V' \subseteq V$ of an undirected graph $G = (V, E)$ is a subset of V satisfying the condition that for any edge $(u, v) \in E$, at least one of u and v is not contained in V'. The maximum independent set problem is the problem of finding an independent set of G with the maximum number of elements.

Suppose that a path decomposition $(X_1, X_2, \ldots)$ of G is obtained, and let w be the pathwidth of the path decomposition. For any $i = 1, 2, 3, \ldots$, consider the following. Let B be an arbitrary node set such that $B \subseteq X_i$. On the subgraph $G[\bigcup_{j\leq i} X_j]$ of G induced by $\bigcup_{j\leq i} X_j$, consider an independent set with the maximum number of nodes that contains all the nodes in B, and let $p_i(B)$ be the number of its elements (if such an independent set does not exist, $p_i(B) = -\infty$). For any node $v \in X_{i+1} \setminus X_i$, there is no edge of E connecting v and any node in $X_{\mathrm{L}}(= \bigcup_{j<i} X_j \setminus X_i)$, by the above property. Therefore, we only need to look at the nodes of B (contained in X_i) to determine whether to add v as an element of the independent set. Although we omit the detail, from the value of $p_i(B)$ for any node set $B \subseteq X_i$, we can calculate the value of $p_{i+1}(B')$ for any node set $B' \subseteq X_{i+1}$, and by performing this calculation for $i = 1, 2, \ldots$ in that order, the maximum independent set problem can be solved. The

information to be stored in each step is the value of $p_i(B)$ for any node set $B \subseteq X_i$, the number of which is at most 2^{w+1}. This algorithm runs in $2^{O(w)}\text{poly}(|V|)$ time, where $\text{poly}(|V|)$ is a polynomial in $|V|$. If w can be regarded as a sufficiently small constant, this can be regarded as the input polynomial time.

We describe path decomposition and pathwidth for directed graphs. For a directed path decomposition of a directed graph $D = (N, A)$, condition (P2) is changed to the following condition (P2′).

(P2′) For any $(u, v) \in A$, there are i, j with $i \le j$ such that $u \in X_i$ and $v \in X_j$.

Note that we replace V in (P1) with N for directed path decomposition.

Directed pathwidth for directed graphs can be defined in the same way as pathwidth for undirected graphs.

The directed pathwidth of the DAG is always zero. This can be shown as follows. The nodes of a DAG $D = (N, A)$ can be ordered in a topological order. Here, a topological order is an order of the nodes of $v_1, \dots, v_{|N|}$ on N such that $i < j$ for any $(v_i, v_j) \in A$. If we construct the bag $X_i = \{v_{|N|-i+1}\}$ for $i = 1, \dots, |N|$, we can confirm that $(X_1, \dots, X_{|N|})$ satisfies (P1), (P2′), and (P3). The directed pathwidth of this directed path decomposition is always 0. Therefore, the directed pathwidth provides no information on the DAG.

7.4.2 DAG-Pathwidth

We explain DAG-path-decomposition (DAG-PD) and DAG-pathwidth [7]. The DAG-PD is defined for a general directed graph $D = (N, A)$. The DAG-PD of $D = (N, A)$ is a sequence $(X_1, X_2, \dots)$ of subsets of N satisfying the following conditions ((D1) is equivalent to (P1) and (D3) to (P3)).

(D1) $\bigcup_{i=1,2,\dots} X_i = N$.
(D2) For every arc $(u, v) \in A$, if $u \in X_1$, $v \in X_1$ holds. Otherwise ($u \notin X_1$), there is an integer $i \ge 2$ such that $u \in X_i$, $u \notin X_{i-1}$ and $v \in X_i$ hold.
(D3) For all i, j, k with $i < j < k$, $X_i \cap X_k \subseteq X_j$ holds.

The condition (D2) is explained below. When $u \notin X_1$, from condition (D1), u must be contained in one of the bags. Therefore, there is always an integer i such that $u \in X_i$ and $u \notin X_{i-1}$. Also, from condition (D3), there exists only one such i. (If there exists i, i' ($i < i'$) such that $u \in X_i$, $u \notin X_{i-1}$, $u \in X_{i'}$, $u \notin X_{i'-1}$, then from (D3) for any j, $i < j < i'$, $X_i \cap X_{i'} \subseteq X_j$, but $u \in X_i \cap X_{i'}$ and $u \notin X_{i'-1}$, a contradiction.) For such i, (D2) requires that the endpoint v of the arc (u, v) is contained in X_i. We write such i, i.e., $\min\{i \mid v \in X_i\}$, as $\mathsf{fb}_X(v)$ or simply $\mathsf{fb}(v)$. fb stands for the First Bag. Intuitively, condition (D2) implies that every endpoint of an arc whose tail is u must be in $X_{\mathsf{fb}(u)}$. This direction condition differs from condition (P2) for undirected graphs.

We have the following lemma on fb.

Lemma 7.1 ([7]) *Let $D = (N, A)$ be a directed graph and $X = (X_1, X_2, \ldots)$ be a DAG-PD of D. Then, for any nodes $u, v \in N$, if there is a directed path from u to v, $\mathsf{fb}(v) \leq \mathsf{fb}(u)$ holds.*

The following lemma can be obtained from Lemma 7.1.

Lemma 7.2 ([7]) *Let $D = (N, A)$ be a directed graph and $X = (X_1, X_2, \ldots)$ be a DAG-PD of D. If two nodes $u, v \in N$ are included in the same strongly connected component, then $\mathsf{fb}(u) = \mathsf{fb}(v)$ holds.*

This lemma holds because u, v are in the same strongly connected component, and thus there exist directed paths from u to v and from v to u.

7.4.3 Nice DAG-PD

Undirected path decomposition has the concept of a nice path decomposition that is useful in dynamic programming. A similar concept can be defined for DAG-pathwidth.

Let $D = (N, A)$ be a directed graph and $X = (X_1, X_2, \ldots)$ be a DAG-PD of D. X is called a *nice* DAG-PD if the following conditions hold:

1. $X_1 = \emptyset$.
2. For $i = 1, 2, \ldots$, exactly one of the following holds:
   - (2-a) (Introduce) There is a strongly connected component S such that $S \cap X_i = \emptyset$ and $X_{i+1} = X_i \cup S$.
   - (2-b) (Forget) There is a node $v \in X_i$ such that $X_{i+1} = X_i \setminus \{v\}$.

A nice DAG-PD starts with $X_1 = \emptyset$. For $i = 1, 2, \ldots$, we consider constructing X_{i+1} from X_i by adding nodes to X_i or deleting nodes from X_i. If we try to add a node $v \in V$ to X_i, all the nodes in the strongly connected component having v must be added to X_i at the same time because the fb values of all the nodes are equal by Lemma 7.2. This operation is called "Introduce." Therefore, (2-a) is defined as above. On the other hand, the bag obtained by deleting any single node from X_i does not violate any conditions of DAG-PD. This operation is called "Forget." Thus, (2-b) is defined as above. In this way, changes in consecutive two bags before and after are minimized to facilitate the design of dynamic programming.

For any DAG-PD, we describe how to change the DAG-PD to a nice DAG-PD without changing its DAG-pathwidth. Let X be the original DAG-PD. Let X_i be the i-th bag of X. First, if the head of X is not $\emptyset$, add $\emptyset$ to the beginning of X. In the following, the bag after the modification is also written as X. Next, when there are two bags such that $X_i = X_{i+1}$, X_{i+1} is removed from X. Finally, repeat the following operations (i), (ii), and (iii) for X as many times as possible.

(i) If there exist bags X_i, X_{i+1} such that $X_i \not\subseteq X_{i+1}$ and $X_{i+1} \not\subseteq X_i$, then insert $X_i \cap X_{i+1}$ between X_i and X_{i+1}. (This results in $X_i \supseteq X_i \cap X_{i+1} \subseteq X_{i+1}$.)
(ii) If there exist bags X_i, X_{i+1} such that $X_{i+1} \subseteq X_i$ and $|X_i| - |X_{i+1}| \geq 2$, insert $X_i \setminus \{v\}$ for some $v \in X_i \setminus X_{i+1}$ between X_i and X_{i+1}.
(iii) If there exist bags X_i, X_{i+1} such that $X_i \subseteq X_{i+1}$ holds, and $X_{i+1} \setminus X_i$ is not a strongly connected component of D, insert $X_{i+1} \setminus A$ between X_i and X_{i+1}, where A is a strongly connected component of $D[X_{i+1} \setminus X_i]$ (subgraph induced by $X_{i+1} \setminus X_i$) such that there is no directed path from the nodes in $(X_{i+1} \setminus X_i \setminus A)$ to the nodes in A.

It is easy to check that the sequence of bags constructed in this way satisfies the conditions for a DAG-PD and a nice DAG-PD. Also, since there are a finite number of nodes in N, the procedure always halts (we omit the proof).

The problem of finding a DAG-PD with minimum DAG-pathwidth is proved to be NP-hard [7]. The proof is based on a reduction from the minimum cut linear arrangement problem, which is NP-hard.

7.5 Discord *k*-Independent Set Problem

In this section, we describe an efficient algorithm for solving the maximum discord k-independent set problem based on DAG-PD [7]. Assume that we are given a DAG $D = (N, A)$ and a nice DAG-PD $X = (X_1, X_2, \ldots, X_r)$ of D as the input of the problem, where r is a positive integer. We design a dynamic programming-based algorithm using X. Recall that when some strongly connected component S is introduced in the DAG-PD, $X_{i+1} = X_i \cup S$ holds. Since D is a DAG, every strongly connected component of D consists of exactly one node. Therefore, in the nice DAG-PD, if $X_{i+1} = X_i \cup S$ for some i and some strongly connected component S, $S = \{v\}$ for some $v \in N$. In this case, we call X_{i+1} an *introduce bag*. If $X_{i+1} = X_i \setminus \{w\}$ for some $w \in N$, we call X_{i+1} a *forget bag*. In those cases, we say that v (resp. w) is a *node of* the bag X_{i+1}.

The process of our algorithm is considered as the construction of a discord k-independent set by deciding whether we add each node to the set or not one by one in the order of the nodes of the introduce bags of the DAG-PD. Let us consider an example shown in Fig. 7.2. An example of an introduce bag $X_{i+1} = X_i \cup \{v\}$ is shown in the figure. We are constructing a discord k-independent set, denoted by Y. Now, we add the node v to Y. Let $Y' = Y \cup \{v\}$ and $Z_{i'} = \bigcup_{j=1,\ldots,i'} X_j$. We need to decide whether the discord condition $\mathsf{discord}(u, Y', D[Z_{i+1}]) = |Y' \cap \mathsf{anticone}(u, D[Z_{i+1}])| \leq k$ holds for all $u \in X_{i+1}$ after adding v to Y. (Recall that $D[Z_{i+1}]$ is the DAG induced by Z_{i+1}.)

First, consider $\mathsf{discord}$ of the newly introduced node v. There is no arc whose tail belongs to Z_{i+1} and whose head is v because for every arc $(w, v) \in A$ for $w \in N$, w first appears in X_j for $j \geq i + 2$ because of conditions (D2) and (D3) (note that v first appears in X_{i+1}). Therefore, there is no node $\bar{w} \in Z_{i+1}$ such that $\bar{w} \rightsquigarrow_{D[Z_{i+1}]} v$.

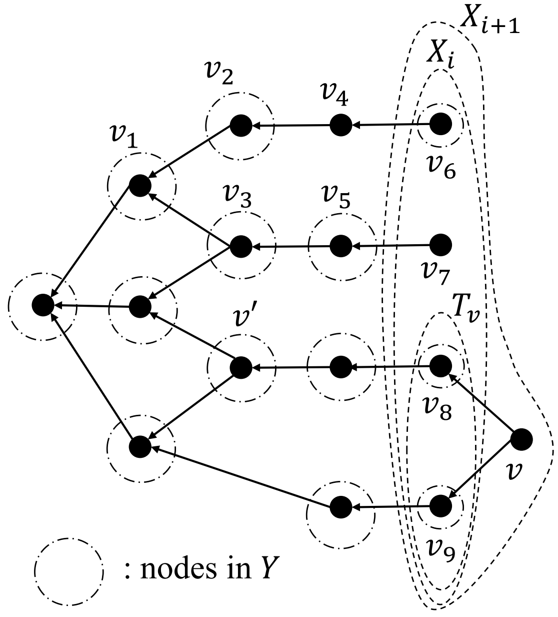


Fig. 7.2 Example of an introduce bag. The DAG shown in the figure is $D[Z_{i+1}] = D[\bigcup_{j=1,\ldots,i+1} X_j]$

Thus, we obtain $\mathsf{discord}(v, Y', D[Z_{i+1}]) = |Y' \cap \mathsf{anticone}(v, D[Z_{i+1}])| = |\{w \in Y' | v \not\rightsquigarrow_{D[Z_{i+1}]} w\}|$; i.e., the number of nodes in Y' that are not reachable from v on $D[Z_{i+1}]$. In the example, v_1, v_2, v_3, v_5 and v_6 are in Y' and are not reachable from v on $D[Z_{i+1}]$, which implies $\mathsf{discord}(v, Y', D[Z_{i+1}]) = 5$.

Next, consider $\mathsf{discord}$ of the nodes in Z_i. Let us observe how $\mathsf{discord}$ of a node in Z_i increases by adding v to Y. As we mentioned above, there is no $\bar{w} \in Z_{i+1}$ such that $\bar{w} \rightsquigarrow_{D[Z_{i+1}]} v$. Therefore, if a node, say v', in Z_i is reachable from v on $D[Z_{i+1}]$, $\mathsf{discord}$ of v' does not increase (see v' in the figure). On the other hand, if a node (e.g., v_1) is not reachable from v on $D[Z_{i+1}]$, $\mathsf{discord}$ of the node increases by one. In the example, $\mathsf{discord}$'s of v_1, v_2, v_3, v_5 and v_6 increase by one and those of the other nodes in Y do not increase.

In our dynamic programming-based algorithm, we do not maintain $\mathsf{discord}$'s of all the nodes. Instead, we maintain the following information on the nodes in X_i when we have processed $X_1, \ldots, X_i$. First, for each subset $S \subseteq X_i$, we consider the set of nodes in Y that are reachable from all the nodes in S and reachable from none of the nodes in $X_i \setminus S$, which we denote by $R(S)$. We define $P(S)$ by

$$P(S) = |R(S)| = |\{v \in Y | \forall u \in S, u \rightsquigarrow_{D[Z_i]} v, \forall w \in X_i \setminus S, w \not\rightsquigarrow_{D[Z_i]} v\}|. \quad (7.1)$$

We maintain $P(S)$ for each subset $S \subseteq X_i$. In the example, $P(\{v_7\}) = |\{v_3, v_5\}| = 2$ because v_3 and v_5 are reachable from v_7 and not reachable from v_6, v_8, v_9 $(\in X_i \setminus \{v_7\})$. Also, $P(\{v_6, v_7\}) = |\{v_1\}| = 1$. By using $P(S)$ for all $S \subseteq X_i$, we can compute $\mathsf{discord}$ of v when we add v to Y as follows:

$$\mathsf{discord}(v, Y', D[Z_{i+1}]) = \left| \sum_{S:S\subseteq X_i, S\cap T_v=\emptyset} P(S) \right|,$$

where T_v is the set of nodes in X_i that are reachable from v. This is because $\mathsf{discord}$ of v is the number of nodes that are not reachable from v, and because any node in $R(S)$ is reachable from v if $S \cap T_v \neq \emptyset$. For example, $\mathsf{discord}(v, Y', D[Z_{i+1}]) = |P(\{v_6, v_7\})| + |P(\{v_6\})| + |P(\{v_7\})| + |P(\emptyset)| = |\{v_1\}| + |\{v_2, v_6\}| + |\{v_3, v_5\}| + 0 = 5$.

Secondly, to compute $\mathsf{discord}$ of the nodes in Y, for each subset $S \subseteq X_i$, we maintain the maximum value of $\mathsf{discord}$ among the nodes in $R(S)$, which we denote by $Q(S)$. We have

$$Q(S) = \max_{u\in R(S)} \{\mathsf{discord}(u, Y, D[Z_i])\}. \tag{7.2}$$

In the example, $Q(\{v_7\}) = \max\{\mathsf{discord}(v_3, Y, D[Z_i]), \mathsf{discord}(v_5, Y, D[Z_i])\} = \max\{8, 8\} = 8$ and $Q(\{v_6, v_7\}) = \max\{\mathsf{discord}(v_1, Y, D[Z_i])\} = 7$. By using $Q(S)$ for all $S \subseteq X_i$, we can decide whether all $\mathsf{discord}$s of the nodes v' in Y are at most k after we add v to Y by checking if $Q(S') + 1 \leq k$ for all subsets $S' \subseteq X_i$ such that $S' \cap T_v = \emptyset$, and by checking if $Q(S'') \leq k$ for all subsets $S'' \subseteq X_i$ such that $S'' \cap T_v \neq \emptyset$ (the latter always holds if we have conducted the check for the previous bags $X_1, \ldots, X_{i-1}$). This is because $\mathsf{discord}$ of a node in Y increases by one if the node is not reachable from v. $S'' \cap T_v \neq \emptyset$ (resp. $S'' \cap T_v = \emptyset$) means that any node in $R(S'')$ is (resp. is not) reachable from v.

A *state* of our dynamic programming-based algorithm is defined as a triple (i, P, Q). The first element i of the state means that the bags $X_1, \ldots, X_i$ are processed. P and Q, described above, are maps from a subset of X_i to an integer. The update of a state (i, P, Q) is given as follows: First, consider the case X_{i+1} is an introduce bag; that is, $X_{i+1} = X_i \cup \{v\}$ for some $v \in N$. In this case, we divide this case into two subcases: (i) v is added to a discord k-independent set. (ii) v is not added to a discord k-independent set. For (i), let $(i + 1, P', Q')$ be the updated state. Then, P' and Q' are given as follows:

$$P'(S) = \begin{cases} 1 & (S = \{v\}), \\ P(S) & (S \cap T_v = \emptyset), \\ P(S \setminus \{v\}) & (S \cap T_v \supsetneq \{v\}), \\ 0 & \text{otherwise}, \end{cases} \tag{7.3}$$

$$Q'(S) = \begin{cases} \sum_{S' : S' \cap T_v = \emptyset} P(S') & (S = \{v\}), \\ Q(S) + 1 & (S \cap T_v = \emptyset \text{ and } P(S) > 0), \\ 0 & (S \cap T_v = \emptyset \text{ and } P(S) = 0), \\ Q(S \setminus \{v\}) & (S \cap T_v \supsetneq \{v\}), \\ 0 & \text{otherwise}, \end{cases} \tag{7.4}$$

for a subset S of X_i. The detail is described in the paper [7]. For (ii), let $(i + 1, P'', Q'')$ be the updated state. Then, P'' and Q'' are given as follows [7]:

$$P''(S) = \begin{cases} 0 & (S = \{v\}), \\ P(S) & (S \cap T_v = \emptyset), \\ P(S \setminus \{v\}) & (S \cap T_v \supsetneq \{v\}), \\ 0 & \text{otherwise}, \end{cases} \tag{7.5}$$

$$Q''(S) = \begin{cases} 0 & (S = \{v\}), \\ Q(S) & (S \cap T_v = \emptyset), \\ Q(S \setminus \{v\}) & (S \cap T_v \supsetneq \{v\}), \\ 0 & \text{otherwise}, \end{cases} \tag{7.6}$$

for a subset S of X_i.

Secondly, we consider (iii) X_{i+1} is a forget bag; that is, $X_{i+1} = X_i \setminus \{v\}$ for some $v \in N$. In this case, let $(i + 1, P''', Q''')$ be the updated state. Then, we obtain the following [7]:

$$P'''(S) = P(S) + P(S \cup \{v\}), \tag{7.7}$$
$$Q'''(S) = \max\{Q(S), Q(S \cup \{v\})\}, \tag{7.8}$$

for a subset S of X_i.

Our algorithm maintains a table table$[i][P][Q]$, where (i, P, Q) is a state. The value of table$[i][P][Q]$ is the maximum number $|Y|$ of discord k-independent sets Y such that P, Q, and Y satisfy Eqs. (7.1) and (7.2). Let P_0 and Q_0 be maps such that $P_0(\emptyset) = Q_0(\emptyset) = 0$ (note that $X_1 = \emptyset$ in the nice DAG-PD, and thus P and Q for X_1 discussed above take only $\emptyset$). The algorithm is given as follows [7]:

1. table$[1][P_0][Q_0] \leftarrow 0$ and table$[i][P][Q] \leftarrow -\infty$ for all $i = 1, \ldots, r$ and all possible P and Q except for $i = 1$, $P = P_0$ and $Q = Q_0$.
2. For $i = 1, 2, \ldots, r - 1$, if X_{i+1} is an introduce bag, do the following:
   (i) For all possible P and Q, let P' and Q' be given by Eqs. (7.3) and (7.4). Then, if $Q'(S) \leq k$ for all $S \subseteq X_{i+1}$, then table$[i + 1][P'][Q'] \leftarrow \max\{$table$[i+1][P'][Q']$, table$[i][P][Q]+1\}$. Otherwise, table$[i + 1][P'][Q'] \leftarrow -\infty$.

(ii) For all possible P and Q, let P'' and Q'' be given by Eqs. (7.5) and (7.6). Then, $\texttt{table}[i+1][P''][Q''] \leftarrow \max\{\texttt{table}[i+1][P''][Q''],$ $\texttt{table}[i][P][Q]\}$.

Otherwise (if X_{i+1} is a forget bag), do the following:

(iii) For all possible P and Q, let P''' and Q''' be given by Eqs. (7.7) and (7.8). Then, $\texttt{table}[i+1][P'''][Q'''] \leftarrow \max\{\texttt{table}[i+1][P'''][Q'''],$ $\texttt{table}[i][P][Q]\}$.

3. Output the maximum value of $\texttt{table}[r][P][Q]$ among all possible P and Q.

We can prove the following theorem using the above algorithm.

Theorem 7.1 ([7]) *Given a DAG $D = (N, A)$ and a nice DAG-PD of D with DAG-pathwidth W, we can solve the maximum discord k-independent set problem in $O((k+1)^{2^{W+1}}(k+2)^{2^{W+1}}2^W W^2|N|)$ time.*

If we can regard k and the DAG-pathwidth W of D as (small) constants, our algorithm can solve the maximum discord k-independent set problem in linear time in $|N|$.

7.6 Conclusion

In this chapter, we provided an overview of the blockchain and introduced pathwidth and directed pathwidth. We introduced DAG-path-decomposition and DAG-pathwidth and described the characterization of DAGs by DAG-pathwidth. We used DAG-path-decomposition to design an algorithm that runs faster when the DAG-pathwidth of the input DAG is small. The concept of DAG-path-decomposition is applicable not only to blockchain DAGs, but also to many DAG structures that appear in the real world, and is a subject for future work. Just as we defined DAG-pathwidth that extends (undirected) pathwidth, we can probably define DAG-treewidth that extends (undirected) treewidth, which is also future work.

References

1. S. Arnborg, D.G. Corneil, A. Proskurowski, Complexity of finding embeddings in a k-tree. SIAM J. Matrix Anal. Appl. **8**(2), 277–284 (1987). https://doi.org/10.1137/0608024
2. S. Arnborg, A. Proskurowski, Linear time algorithms for NP-hard problems restricted to partial k-trees. Discrete Appl. Math. **23**(1), 11–24 (1989)
3. D. Berwanger, A. Dawar, P. Hunter, S. Kreutzer, J. Obdržálek, The dag-width of directed graphs. J. Comb. Theory Ser. B **102**(4), 900–923 (2012)
4. Y. Chen, Y. Chen, An efficient algorithm for answering graph reachability queries, in *IEEE 24th International Conference on Data Engineering* (2008), pp. 893–902
5. R. Halin, S-functions for graphs. J. Geom. **8**(1), 171–186 (1976)

6. T. Johnson, N. Robertson, P. Seymour, R. Thomas, Directed tree-width. J. Comb. Theory Ser. B **82**(1), 138–154 (2001)
7. S. Kasahara, J. Kawahara, S. Minato, J. Mori, DAG-pathwidth: graph algorithmic analyses of DAG-type blockchain networks. IEICE Trans. Inf. Syst. **E106.D**(3), 272–283 (2023)
8. Y. Lewenberg, Y. Sompolinsky, A. Zohar, Inclusive block chain protocols, in *Financial Cryptography and Data Security* (2015), pp. 528–547
9. S. Nakamoto, Bitcoin: a peer-to-peer electronic cash system (2009)
10. B.A. Reed, Tree width and tangles: a new connectivity measure and some applications, in *Surveys in Combinatorics* (Cambridge University Press, 1997), pp. 87–162
11. N. Robertson, P. Seymour, Graph minors. I. Excluding a forest. J. Comb. Theory Ser. B **35**(1), 39–61 (1983)
12. N. Robertson, P. Seymour, Graph minors. III. Planar tree-width. J. Comb. Theory Ser. B **36**(1), 49–64 (1984)
13. Y. Sompolinsky, A. Zohar, PHANTOM: a scalable blockDAG protocol. IACR Cryptol. ePrint Archive **2018**, 104 (2018)

Part IV
Algorithms for Mobility Society

Chapter 8
System-Control-Based Approach to Car-Sharing Systems

Kazunori Sakurama, Kenji Kashima, Takuya Ikeda, Naoki Hayashi, Kenta Hoshino, Masaki Ogura, and Chengyan Zhao

8.1 Introduction

The development of information and communication technologies (ICT) and data science is changing the structure of society, including the movement of people and goods or mobility. To this day, various studies have been conducted to innovate mobility. Mobility-as-a-service (MaaS) is attracting attention and is being adopted worldwide. As an exemplar case, Whim in Finland [9, 30] provides a transportation package that converts road transportation and railroads into on-demand services.

K. Sakurama (✉) · K. Kashima · K. Hoshino
Kyoto University, Yoshida-honmachi, Sakyo-ku, Kyoto 606-8501, Japan
e-mail: sakurama@i.kyoto-u.ac.jp

K. Kashima
e-mail: kk@i.kyoto-u.ac.jp

K. Hoshino
e-mail: hoshino@i.kyoto-u.ac.jp

T. Ikeda
The University of Kitakyushu, 1-1 Hibikino, Wakamatsu-ku, Kitakyushu, Fukuoka 808-0135, Japan
e-mail: t-ikeda@kitakyu-u.ac.jp

N. Hayashi
Osaka University, 1-3 Machikaneyama-cho, Toyonaka, Osaka 560-8531, Japan
e-mail: n.hayashi@sys.es.osaka-u.ac.jp

M. Ogura
Osaka University, 1-5 Yamadaoka, Suita, Osaka 565-0871, Japan
e-mail: m-ogura@ist.osaka-u.ac.jp

C. Zhao
Ritsumeikan University, 1-1-1 Nojihigashi, Kusatsu, Shiga 525-0058, Japan
e-mail: c-zhao@fc.ritsumei.ac.jp

K. Ikeda et al. (eds.), *Advanced Mathematical Science for Mobility Society*,
https://doi.org/10.1007/978-981-99-9772-5_8

This chapter focuses on car-sharing services, particularly one-way car-sharing services, as one of the forms of MaaS. One-way car-sharing services have multiple stations where vehicles are pooled, and users can return their vehicles to any of the available stations (i.e., parking lots). The service is expected to grow because of its high convenience to users and the ability to use ordinary parking lots as stations [51]. In Japan, government approval for these services was granted in 2014, and experimental operations are underway [55], including Ha:mo Ride TOYOTA.

However, the current one-way car-sharing service poses a significant challenge to the operator, the most pertinent of which is the uneven distribution of vehicles. Thus, research is being conducted to solve these problems [41]. The uneven distribution of vehicles refers to the situation wherein some stations run out of parked vehicles whereas others run out of parking spaces due to asymmetrical user demand. An obvious and straightforward solution is for service staff to relocate vehicles to redistribute the number of vehicles among stations. However, this approach carries high labor costs of vehicle relocation (i.e., requiring provider staff to drive vehicles between stations); thus, cost-effective methods need to be considered, which has attracted considerable research attention. For example, in [5], a method to minimize the cost of relocation using the model predictive control method was proposed. Moreover, in [69], a model that considers staff behavior in relocation was developed. In [3], a study was conducted to minimize the number of relocations for a car-sharing system in a resort area in Southern California.

This chapter discusses control system approaches to overcome the aforementioned issues. Thus, recent research trends are presented that show that a systems control approach is promising for mobility innovation. Section 8.2 provides the result of the optimization of the car-sharing system considering demand shift by dynamic pricing. Section 8.3 presents sparse optimal control for networked systems. In Sect. 8.4, communication-aware cooperative car-sharing control is discussed. Section 8.5 considers the rebalancing in car-sharing services as a control problem of probability distributions with costs determined by the Wasserstein distance. Section 8.6 presents the optimization of one-way car-sharing services via the DC program.

Notation

Let $\mathbb{R}$, $\mathbb{R}_+$, and $\mathbb{R}_{++}$ denote the set of real, non-negative, and positive numbers, respectively. Let $\mathbb{Z}$ and $\mathbb{Z}_+$ be the sets of integers and non-negative integers, respectively. The set of corresponding vectors of size n are denoted by $\mathbb{R}^n$, $\mathbb{R}^n_+$, and $\mathbb{R}^n_{++}$, respectively. We let 1_n denote a column vector with all entries set to unity. The identity and zero matrices of order n are denoted by I_n and O_n, respectively. For real numbers $a_1, \dots, a_n \in \mathbb{R}$ or a vector $a = [a_1 \ \cdots \ a_n]^\top \in \mathbb{R}^n$, $\mathrm{diag}(a_1, \dots, a_n)$ represents the diagonal matrix whose i-th entry is a_i.

For a real number $\kappa \in \mathbb{R}$, let $\kappa\mathbb{Z} = \{\dots, -2\kappa, -\kappa, 0, \kappa, 2\kappa, \dots\}$. The floor and ceiling functions for κ are defined as

$$\lfloor x \rfloor_\kappa = \max\{y \in \kappa\mathbb{Z} : y \le x\},\ \lceil x \rceil_\kappa = \min\{y \in \kappa\mathbb{Z} : y \ge x\},$$

respectively. Consider a random variable $x : \mathbb{Z}_+ \to \mathbb{Z}_+$ of a non-negative integer at time $t \in \mathbb{Z}_+$. The probability of $x(t)$ being $x \in \mathbb{Z}_+$ is denoted as $\mathrm{P}(x(t) = x) \in [0, 1]$. The expectation of a function $f : \mathbb{Z}_+ \to \mathbb{R}$ of $x(t)$, $f(x(t))$, is expressed as

$$\mathrm{E}[f(x(t))] = \sum_{x=0}^{\infty} f(x)\mathrm{P}(x(t) = x).$$

For random variables $x, y : \mathbb{Z}_+ \to \mathbb{Z}_+$, the joint probability of $x(t)$, $y(t)$ being $x, y \in \mathbb{Z}_+$ is denoted as $\mathrm{P}(x(t) = x, y(t) = y) \in [0, 1]$. The conditional probability of $x(t)$ being $x \in \mathbb{Z}_+$ under the assumption that $y(t)$ takes $y \in \mathbb{Z}_+$ is represented as $\mathrm{P}(x(t) = x|y(t) = y)$, which is defined as

$$\mathrm{P}(x(t) = x|y(t) = y) = \frac{\mathrm{P}(x(t) = x, y(t) = y)}{\mathrm{P}(y(t) = y)}.$$

For a function $f : \mathbb{Z}_+ \to \mathbb{R}$ of a variable $x(t)$, the expectation of $f(x(t))$ under the assumption that $y(t)$ takes $y \in \mathbb{Z}_+$ is defined as

$$\mathrm{E}[f(x(t))|y(t) = y] = \sum_{x=0}^{\infty} f(x)\mathrm{P}(x(t) = x|y(t) = y).$$

For some natural number m and set $\Omega \subset \mathbb{R}$, $a \in \Omega^m$ implies that $a_i \in \Omega$ for any i, where $a = [a_1, a_2, \ldots, a_m]^\top \in \mathbb{R}^m$. Let

$$\|a\|_{\ell^0} \triangleq \#\{j \in \{1, 2, \ldots, m\} : a_j \neq 0\},$$
$$\|a\|_{\ell^1} \triangleq \sum_{j=1}^{m} |a_j|$$

denote ℓ^0, ℓ^1 norm of vector a, where # returns the number of elements of a set. For the function

$$u(t) = [u_1(t), u_2(t), \ldots, u_m(t)]^\top \in \mathbb{R}^m$$

on a time interval $[0, T]$, let

$$\|u\|_{L^0} \triangleq \sum_{j=1}^{m} \mu(\{t \in [0, T] : u_j(t) \neq 0\}),$$
$$\|u\|_{L^1} \triangleq \sum_{j=1}^{m} \int_0^T |u_j(t)|dt$$

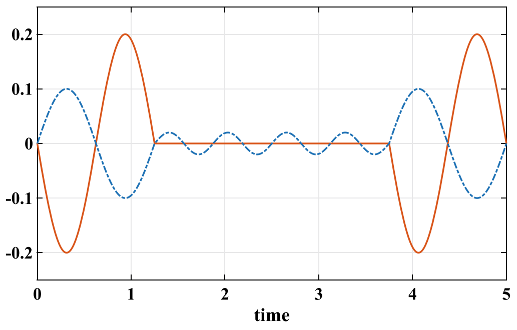


Fig. 8.1 Sparsity of signal

denote its L^0, L^1 norm, where μ is the length in the naive sense (Lebesgue measure in $\mathbb{R}$). Thus, the L^0 norm of the signals in Fig. 8.1 is 5 for the blue one, and approximately 2.6 for the red one. Notably, [0, 1] denotes a set of real number between 0 and 1, but {0, 1} denotes a set of binary value 0 and 1.

The real matrix A is said to be non-negative (positive), and is denoted by $A \geq 0$ ($A > 0$), if all entries of A are non-negative (positive). The notion $B < A$ is defined as $B - A < 0$. Let the Hadamard product of matrices A and B be denoted by $A \odot B$. Let $A \otimes B$ denote the Kronecker product of matrices A and B. If A and B are square, the Kronecker sum of A and B is defined as $A \oplus B = A \otimes I_m + I_n \otimes B$, where n and m denote the orders of A and B, respectively. We define the entry-wise exponential operation of a real vector v as $\exp[v] = [\exp v_1, \ldots, \exp v_n]^\top$ and the entry-wise logarithm operation as $\log[v] = [\log v_1, \ldots, \log v_n]^\top$. For a vector v with scalar entries $v_1, \ldots, v_n$, we use $\mathrm{diag}(v_1, \ldots, v_n)$ to denote the diagonal matrix.

8.2 Optimization of the Car-Sharing System Considering Demand Shift by Dynamic Pricing

To solve the uneven distribution of vehicles in car-sharing services, we consider introducing dynamic prices. This is a method of voluntarily moving users so that the vehicles are evenly distributed by adjusting the prices of the service hourly. From this aspect, [45] proposes a model that determines the price to maximize the profits of both the user and the operator. The model in [75] aims to maximize profits by considering the allocation of relocating personnel. Reference [74] investigated optimal pricing and charging scheduling for an electric vehicle sharing system. In addition, [47, 72] examined pricing strategies for a one-way car-sharing service. Most of the models presented in these papers only consider that demand increases or decreases with changes in prices. However, these models do not adequately represent user behavior. Users will probably prefer and shift to less expensive links (i.e., change their starting and ending destinations on foot). Such a shift is particularly likely to occur in urban areas, where many stations are within walking distance of each other. The authors

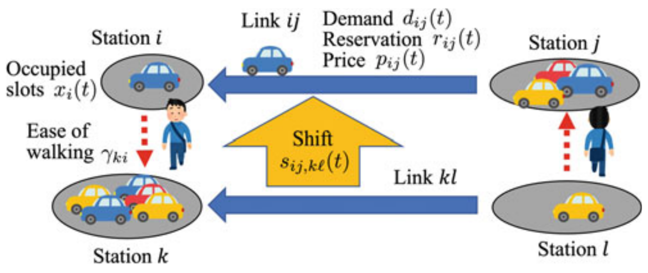


Fig. 8.2 Model of car-sharing system [66]

proposed a demand shift model and a dynamic pricing model to represent such user behavior in [70], and addressed the optimal control problem of the traffic model in [66, 67]. In this section, the research results in [66, 67] will be presented.

8.2.1 Modeling

The car-sharing service considered here has dynamic prices, and users can reserve a car by referring to the prices via the Internet or other means. Prices are changed at regular intervals according to a policy determined by the organizer. The user makes a reservation by declaring both the departure and arrival stations.

The number of stations is $n \in \mathbb{Z}_+$. Let $\mathcal{N} = \{1, 2, \ldots, n\}$ be the set of station indices. A move whose departure station is $j \in \mathcal{N}$ and whose arrival station is $i \in \mathcal{N}$ is called a link ij. The target car-sharing system consists of four models: the number of parking spaces, the number of reservations, the number of demands, and the number of demand shifts. Figure 8.2 shows the overall diagram.

8.2.1.1 Model of the Number of Parking Spaces

Consider station $i \in \mathcal{N}$. Let $x_i(t) \in \mathbb{Z}_+$ denote the sum of the number of parking spaces already parked and spaces scheduled to be parked due to reservations. Denote the capacity of parking spaces by $x_i^{\max} \in \mathbb{Z}_+$. In other words, $x_i(t)$ must satisfy

$$0 \le x_i(t) \le x_i^{\max} \ \forall t \in \mathbb{Z}_+. \tag{8.1}$$

Let $m \in \mathbb{Z}_+$ be the total number of vehicles. The sum of the numbers of parked cars $x_i(t)$ at all stations satisfies

$$\sum_{i \in \mathcal{N}} x_i(t) = m. \tag{8.2}$$

We assume that the total capacity of parking spaces

$$m^{\max} = \sum_{i \in \mathcal{N}} x_i^{\max} \tag{8.3}$$

is sufficiently larger than the number of vehicles m.

Denote the number of reservations for link ij by $r_{ij}(t) \in \mathbb{Z}_+$. Then, $x_i(t)$ varies according to $r_{ij}(t)$ as

$$x_i(t+1) = x_i(t) + \sum_{j \in \mathcal{N}} (r_{ij}(t) - r_{ji}(t)). \tag{8.4}$$

In this equation, $r_{ij}(t)$ ($r_{ji}(t)$) denotes the number of vehicles entering (leaving) station i.

8.2.1.2 Model of Reservation Number

Next, consider a model for the reservation number $r_{ij}(t)$. This model is assumed to follow three rules: (i) The reservation number is smaller than the demand number $d_{ij}(t) \in \mathbb{Z}_+$, i.e., $r_{ij}(t) \le d_{ij}(t)$ is satisfied. (ii) The number of parking spaces $x_i(t+1)$ at the next time satisfies condition (8.1). (iii) The unrealized demand $u_{ij}(t) = d_{ij}(t) - r_{ij}(t) \in \mathbb{Z}_+$ is minimized through the dynamic pricing of the organizer.

Here, if service is provided on a first-come, first-serve basis, (iii) cannot be strictly satisfied. However, if the parking capacity is satisfied for demand, i.e.,

$$0 \le x_i^+(t) \le x_i^{\max},\ 0 \le x_j^+(t) \le x_j^{\max} \tag{8.5}$$

then $u_{ij}(t) = 0$, i.e., $r_{ij}(t) = d_{ij}(t)$, is satisfied, where

$$x_i^+(t) = x_i(t) + \sum_{j \in \mathcal{N}} (d_{ij}(t) - d_{ji}(t)). \tag{8.6}$$

In such a case, even if we take the demand as the number of reservations as it is (i.e., $r_{ij}(t) = d_{ij}(t)$), (8.1) is satisfied because $x_i(t+1)$ in (8.4) is equal to $x_i^+(t)$.

From this perspective, to satisfy (iii), we consider controlling the demand number $d_{ij}(t)$ so that $x_i^+(t)$ satisfies (8.5).

8.2.1.3 Model of Demand Number

Consider a model of the demand number $d_{ij}(t)$ for link ij. Assume that users move to nearby stations on foot according to the link's price. The number of demands to shift from link $k\ell$ to ij is denoted by $s_{ij,k\ell}(t) \in \mathbb{Z}_+$. This means that the departure station is moved from ℓ to j and the arrival station is moved from k to i. Denote the original demand under a specific standard price by $\hat{d}_{ij}(t) \in \mathbb{Z}_+$, which is assumed to be a random variable with expected value $\delta_{ij} > 0$, i.e.,

$$\mathrm{E}[\hat{d}_{ij}(t)] = \delta_{ij}. \tag{8.7}$$

A typical example is a Poisson process. Suppose that $\hat{d}_{ij}(t)$ is independent for any $i, j \in \mathcal{N}$ and $t \in \mathbb{Z}_+$. In this case, the demand number $d_{ij}(t)$ changes from the original number $\hat{d}_{ij}(t)$ according to the demand shift number $s_{ij,k\ell}(t)$ as

$$d_{ij}(t) = \hat{d}_{ij}(t) + \sum_{k\ell \in \mathcal{N}^2} (s_{ij,k\ell}(t) - s_{k\ell,ij}(t)). \tag{8.8}$$

8.2.1.4 Model of Demand Shift Number

Next, we model the variation of the demand shift number $s_{ij,k\ell}(t)$ with the price. Denote the price for link ij by $p_{ij}(t) \in \kappa\mathbb{Z}$, where $\kappa \in \mathbb{Z}_+$ ($\kappa > 0$) denotes the unit of the price in this service (e.g., 10 yen, 1 dollar, etc.). The ease of walking between stations i and k is denoted by $\gamma_{ik} (= \gamma_{ki}) \geq 0$. In general, the larger the distance between stations i and k, the smaller is the value of γ_{ik}. Two pieces of information are provided to the users. (i) The price difference between the two links, $p_{k\ell}(t) - p_{ij}(t)$ (the benefit of travel). (ii) Ease of walking $\gamma_{ik}, \gamma_{j\ell}$ (the cost of travel). Based on this information, we assume that demand moves stochastically based on the expectation

$$\mathrm{E}[s_{ij,k\ell}(t) | P(t) = P] = \gamma_{ik}\gamma_{j\ell}\phi(p_{ij} - p_{k\ell}), \tag{8.9}$$

where $P(t)$ and $P \in \mathbb{Z}^{n \times n}$ denote the matrices with $p_{ij}(t)$ and p_{ij} as the (i, j)-component, respectively. The function $\phi : \mathbb{Z} \to \mathbb{R}_+$ is a monotonically non-increasing function satisfying $\phi(p) = 0$ for $p \geq 0$. This indicates that users are more likely to move to links with lower prices and never move to links with higher prices.

8.2.2 Control Objective

The control objective of the organizer is to evenly redistribute the number of vehicles parked at each station to match the unrealized demand $u_{ij}(t)$, minimizing the difference with actual availability. As described in Sect. 8.2.1.2, this is achieved by controlling $x_i^+(t)$ so that the inequality in (8.5) is satisfied. Here, as a soft constraint corresponding to (8.5), the deviation from half of the capacity of parking spaces $\bar{x}_i^+(t) = x_i^+(t) - x_i^{\max}/2$ is considered as a penalty. The goal here is to level this penalty across all stations. We consider the sum of the expected values as the evaluation function

$$V_{\mathrm{x}}(x(t)) = \frac{1}{n} \sum_{i \in \mathcal{N}} \mathrm{E}\left[\bar{x}_i^+(t) - \frac{1}{n} \sum_{i \in \mathcal{N}} \bar{x}_i^+(t)\right]^2, \tag{8.10}$$

where $x(t) = [x_1(t)\ x_2(t)\ \cdots\ x_n(t)]^\top$ and $x_i^+(t)$ is given by $x_i(t)$ as (8.6).

Here, to redistribute the number of parking spaces, the organizer introduces dynamic prices based on a common decentralized policy. The common decentralized policy is based on a common algorithm for all the links using only information from neighboring stations to determine prices for each link. Because this policy does not require information aggregation, it can be implemented regardless of the number of stations. Let $\hat{p}_{ij} = \hat{p}_{ji} \in \kappa\mathbb{Z}$ be the standard price for travel between stations i and j. The dynamic price $p_{ij}(t)$ is determined according to the policy $\pi : \mathbb{Z}_+^2 \to \kappa\mathbb{Z}$ as

$$p_{ij}(t) = \hat{p}_{ij} + \pi(\bar{x}_i(t), \bar{x}_j(t)), \tag{8.11}$$

where $\bar{x}_i(t) = x_i(t) - x_i^{\max}/2$.

For user convenience, the price $p_{ij}(t)$ should be as close as possible to the standard price $\hat{p}_{ij}$. From (8.11), this can be evaluated by

$$V_{\mathrm{p}}(x(t)) = \frac{1}{n^2} \sum_{i,j \in \mathcal{N}} \mathrm{E}[\pi(\bar{x}_i(t), \bar{x}_j(t))]^2. \tag{8.12}$$

Here, for simplicity, $\pi : \mathbb{Z}_+^2 \to \kappa\mathbb{Z}$ is assumed to be an affine function as

$$\pi(\bar{x}_i, \bar{x}_j) = \hat{\pi}_{\mathrm{a}}\bar{x}_i + \hat{\pi}_{\mathrm{b}}\bar{x}_j + \hat{\pi}_{\mathrm{c}},\ \ \hat{\pi}_{\mathrm{a}}, \hat{\pi}_{\mathrm{b}}, \hat{\pi}_{\mathrm{c}} \in \kappa\mathbb{Z}, \tag{8.13}$$

where $\hat{\pi}_{\mathrm{a}}, \hat{\pi}_{\mathrm{b}}, \hat{\pi}_{\mathrm{c}} \in \kappa\mathbb{Z}$ are design parameters.

Consider designing a common decentralized policy π such that the functions $V_{\mathrm{x}}(x(t))$ and $V_{\mathrm{p}}(x(t))$ are reduced. To take these terminal values into account, we employ the evaluation function

$$V(\hat{\pi}_{\mathrm{a}}, \hat{\pi}_{\mathrm{b}}, \hat{\pi}_{\mathrm{c}}) = \max_{m \in \mathbb{Z}} \lim_{t \to \infty} (V_{\mathrm{x}}(\bar{x}(t)) + \mu V_{\mathrm{p}}(x(t))) + \nu(\hat{\pi}_{\mathrm{a}}^2 + \hat{\pi}_{\mathrm{b}}^2 + \hat{\pi}_{\mathrm{c}}^2), \tag{8.14}$$

where $\mu, \nu > 0$ are some constants and the last term is the normalization term. Maximization with respect to $m \in \mathbb{Z}$ is considered to render the policy π scalable independently of the number of vehicles m.

From the above, we consider the following problem.

Problem 8.1 Consider the system consisting of (8.1)–(8.9). Find the solution to the following optimization problem as the parameters of the common decentralized policy $\pi : \mathbb{Z}_+^2 \to \kappa\mathbb{Z}$ in (8.11) that has the form of (8.13).

$$\min_{\hat{\pi}_{\mathrm{a}}, \hat{\pi}_{\mathrm{b}}, \hat{\pi}_{\mathrm{c}} \in \kappa\mathbb{Z}} V(\hat{\pi}_{\mathrm{a}}, \hat{\pi}_{\mathrm{b}}, \hat{\pi}_{\mathrm{c}}) \tag{8.15}$$

8.2.3 Main Result

To obtain an analytical solution to Problem 8.1, we consider a linear system near an equilibrium point. This yields the following theorem.

Theorem 8.1 *Consider the system that linearizes (8.1)–(8.9) under the assumption that the function ϕ satisfies $\mathrm{d}\phi/\mathrm{d}p(p_*) = -\hat{\phi}$ at an equilibrium point $p_* < 0$ for a constant $\hat{\phi} > 0$. Let $x_i(t) \in \mathbb{Z}_+$ $(i \in \mathcal{N})$ be the solution of this system, and let $\Delta, \Gamma \in \mathbb{R}^{n\times n}$ be the matrices whose (i, j)-components are δ_{ij}, γ_{ij}, respectively. Assume that the graph whose adjacent matrix is given by Γ is connected. For the matrix $L = \mathrm{diag}(\Gamma 1_n) - \Gamma \in \mathbb{R}^{n\times n}$, let $\lambda_1 = 0, \lambda_2, \ldots, \lambda_n (0 < \lambda_2 \leq \cdots \leq \lambda_n)$ be its eigenvalues and let $v_1 = 1_n/\sqrt{n}, v_2, \ldots, v_n \in \mathbb{R}^n$ be the corresponding eigenvectors. Then, the parameters $\hat{\pi}_\mathrm{a}, \hat{\pi}_\mathrm{b}, \hat{\pi}_\mathrm{c} \in \kappa\mathbb{Z}$ of the solution to the optimization problem (8.15) in $\pi : \mathbb{Z}_+^2 \to \kappa\mathbb{Z}$ of the dynamic pricing policy (8.11) with (8.13) satisfy*

$$\hat{\pi}_\mathrm{a} \in \left\{ \left\lfloor \sqrt{\frac{\|h\|}{2\sqrt{2n\nu}}} \right\rfloor_\kappa, \left\lceil \sqrt{\frac{\|h\|}{2\sqrt{2n\nu}}} \right\rceil_\kappa, \left\lfloor \frac{1}{\lambda_n \sum_{i,j\in\mathcal{N}} \gamma_{ij}} \right\rfloor_\kappa \right\} \tag{8.16}$$

$$\hat{\pi}_\mathrm{a} \leq \left\lfloor \frac{1}{\lambda_n \sum_{i,j\in\mathcal{N}} \gamma_{ij}} \right\rfloor_\kappa \tag{8.17}$$

$$\hat{\pi}_\mathrm{b} = -\hat{\pi}_\mathrm{a} \tag{8.18}$$

$$\hat{\pi}_\mathrm{c} = 0, \tag{8.19}$$

where h is given as

$$h = \frac{\sum_{i\in\{2,\ldots,n\}} \lambda_i^{-1} v_i v_i^\top (\Delta - \Delta^\top) 1_n}{\hat{\phi} \sum_{i,j\in\mathcal{N}} \gamma_{ij}}. \tag{8.20}$$

The minimum value of (8.15) is derived as

$$\min_{\hat{\pi}_\mathrm{a},\hat{\pi}_\mathrm{b},\hat{\pi}_\mathrm{c}\in\kappa\mathbb{Z}} V(\hat{\pi}_\mathrm{a}, \hat{\pi}_\mathrm{b}, \hat{\pi}_\mathrm{c}) = \frac{1}{n}\frac{\|h\|^2}{4\hat{\pi}_\mathrm{a}^2} + 2\nu\hat{\pi}_\mathrm{a}^2. \tag{8.21}$$

Proof From (8.4), (8.7), (8.8), (8.11), and (8.13), the expectation of $\bar{x}_i(t) = x_i(t) - x_i^{\max}/2$ is governed by the differential equation

$$\mathrm{E}[\bar{x}(t+1)] = (I_n - (\hat{\pi}_\mathrm{a} - \hat{\pi}_\mathrm{b})\hat{\phi} 1_n^\top \Gamma 1_n L)\mathrm{E}[\bar{x}(t)] + (\Delta - \Delta^\top) 1_n, \tag{8.22}$$

where $\bar{x}(t) = [\bar{x}_1(t) \ \cdots \ \bar{x}_n(t)]^\top \in \mathbb{R}^n$. Equation (8.22) is equivalent to a system under consensus control with external input, and its properties can be analyzed from the graph Laplacian L by control theory based on algebraic graph theory [53]. The detailed proof is omitted for reasons of space limitation. □

Equation (8.16) gives three candidates for $\hat{\pi}_\mathrm{a}$. If either of the first two terms is smaller than the last term, then this term is the solution. If both of the first two terms are smaller than the last term, then the solution is to substitute the two terms into (8.21) and output the smaller value. Otherwise, the last term is the solution. Equations (8.18) and (8.19) indicate that the skew-symmetric policy (i.e., $\pi(\bar{x}_i, \bar{x}_j) = -\pi(\bar{x}_j, barx_i)$) is the most effective.

8.2.4 Simulation Result

To verify the effectiveness of the proposed method, a simulation is performed using the traffic simulator SOUND [34]. Figure 8.3 represents a scene in the simulation. The $n = 25$ stations are scattered over an area of 6×8 km, and both vehicles under sharing service and regular vehicles are traveling.

The parameters of the system are as follows. All parking spaces have a capacity of $x_i^{\max} = 10$ vehicles. The number of sharing vehicles is $m = 164$. The time interval for changing prices is set to 5 s, which is the sampling time of the model. We set the price elasticity $\hat{\phi} = 2.5 \times 10^{-5}$, the expected value of demand $\delta_{ij} = 0.005 \times 60^{-1}$–$3 \times 60^{-1}$, and the ease of shift between stations $\gamma_{ik} = e^{-\eta\|\rho_i - \rho_k\|}$, where $\rho_i \in \mathbb{R}^2$ is the

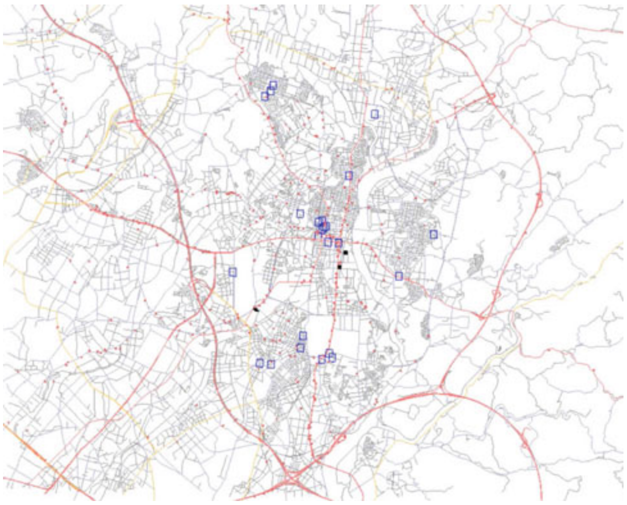


Fig. 8.3 Simulation via SOUND: 25 stations (hollowed-out rectangle), vehicles under sharing service (solid squares), general vehicles (dots) [67]

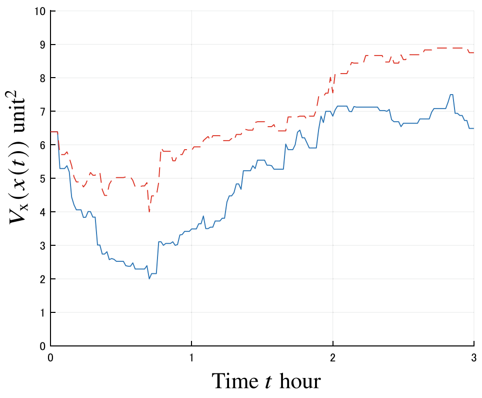


Fig. 8.4 Penalty $V_x(x(t))$ for the number of parking vehicles: (A) dynamic pricing (solid line) and (B) without it (chain line) [67]

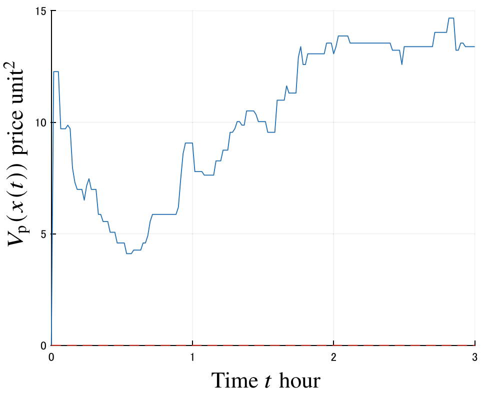


Fig. 8.5 Deviation $V_p(x(t))$ of prices from standard ones: (A) with dynamic pricing (solid line), (B) without it (chain line) [67]

position of station i in Fig. 8.3 and $\eta = 4.5 \times 10^{-4}$. The common decentralized policy $\pi : \mathbb{Z}_+^2 \to \kappa\mathbb{Z}$ in (8.11) is given as in (8.13). Designing its parameters according to Theorem 8.1, we obtain $\hat{\pi}_a = -\hat{\pi}_b = 1$ and $\hat{\pi}_c = 0$ for $\mu = \nu = 0.01$ and $\kappa = 1$ from (8.16), (8.18), and (8.19). Simulations are performed (A) with dynamic pricing and (B) without it for comparison.

Figure 8.4 shows the penalty $V_x(x(t))$ for the number of parking vehicles given by (8.10). Figure 8.5 represents the deviation $V_p(x(t))$ of prices from standard ones given by (8.12). In both cases, the solid lines represent the results (A) with dynamic pricing and the dashed lines represent the results (B) without it. Figure 8.4 shows that the penalty is smaller and the number of parking spaces is smoothed for (A) with dynamic pricing than (B) without it. Figure 8.6 denotes the cumulative total of unfulfilled demand $\sum_{ij} u_{ij}(t)$. It can be seen that (A) with dynamic pricing has fewer unfulfilled demands and more services are provided than (B) without it.

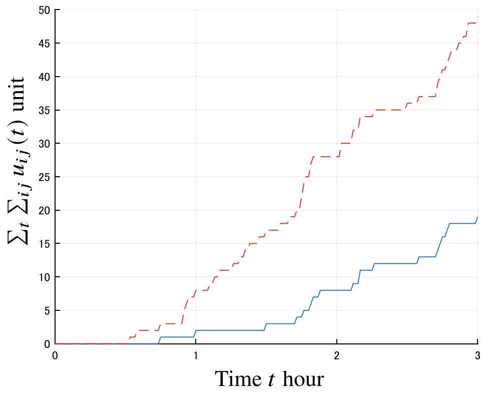


Fig. 8.6 Cumulative unrealized demand $\sum_t \sum_{ij} u_{ij}(t)$: (A) with dynamic pricing (solid line), (B) without it (chain line) [67]

8.3 Sparse Optimal Control for Networked Systems

A signal that takes the value 0 at most times and elements, such as the solid line in Fig. 8.1, is said to be sparse. In recent years, the effectiveness of sparsity-based methods in machine learning, such as regularization or feature extraction, has been attracting considerable research attention [23]. In control engineering, achieving the control goal with sparse input signals is also an important application problem [56]. The authors recently clarified the theoretical properties of sparse optimal control [36, 39, 42] and developed a series of studies on a new problem setting for networked systems applications [35, 37, 38, 40, 60]. The main objective of this section is to apply these results to one-way car sharing services [19, 41].

8.3.1 Sparse Optimal Control Problem

Here we introduce the sparse optimal control problem and show the validity properties of its convex relaxation.

8.3.1.1 Formulation

In this section, we formulate several sparse optimal control problems treated in this chapter. The first is the problem of finding sparse control inputs as described in the introduction [40].

Problem 8.2 For given $A(t) \in \mathbb{R}^{n\times n}$, $B(t) \in \mathbb{R}^{n\times m}$, $T > 0$, x_0, $x_d \in \mathbb{R}^n$, $\beta \in \{1, 2, \ldots, m-1\}$, find the solution to the following optimization problem:

$$\begin{aligned}\underset{u}{\text{minimize}}\ & \|u\|_{L^0}\\ \text{subject to}\ & \dot{x}(t) = A(t)x(t) + B(t)u(t), \quad \forall t \in [0, T],\\ & x(0) = x_0, \quad x(T) = x_d,\\ & \|u(t)\|_{\ell^0} \le \beta, \quad \forall t \in [0, T],\\ & u(t) \in [0, 1]^m, \quad \forall t \in [0, T].\end{aligned}$$

$\lhd$

This problem is to minimize the sum of the time durations over which the inputs are non-zero under the constraint that the number of elements that can be input at each time in a linear time-varying system is less than or equal to β. Note that the box constraint on the input signal (although the upper and lower bounds can be other values) is an essential assumption from which the theoretical results are derived.

The next problem is to maximize the controllability of the linear system [35].

Problem 8.3 For given $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, β, $T > 0$, define the matrix

$$G_c := \int_0^T \exp^{A(T-\tau)} BZ(\tau)B^\top \exp^{A^\top(T-\tau)} d\tau.$$

In this case, find the solution to the following optimization problem:

$$\begin{aligned}\underset{z}{\text{maximize}}\ & \mathrm{Tr}(G_c)\\ \text{subject to}\ & Z(t) = \mathrm{diag}(z(t)), \quad z(t) \in \{0, 1\}^m,\\ & \|z\|_{L^0} \le \beta.\end{aligned}$$

$\lhd$

In this problem, note that $Z(\tau)Z(\tau)^\top = Z(\tau)$ holds and the matrix G_c is the controllability Gramian corresponding to the continuous-time linear system

$$\dot{x}(t) = Ax(t) + BZ(t)u(t). \tag{8.23}$$

The larger[1] the controllability Gramian, the smaller the control inputs required for state transition. However, in the case of $z_i(t) = 0$, the ith input channel is inactive at time t and, therefore, unavailable. In other words, this is a scheduling problem, i.e., which actuator should be active at which time to maximize controllability.

In the last problem, decision variables are included in the A-matrix [37].

Problem 8.4 For given $\mathcal{A}(t)$, $M_e(t) \in \mathbb{R}^{n\times n}$, $e = 1, \ldots, m$, $\alpha \in \mathbb{R}$, $T > 0$, x_0, $x_d \in \mathbb{R}^n$, find the solution to the following optimization problem:

[1] The magnitude of the semi-positive definite matrix is formulated using traces here, but results for other indices are also given in [38, 60].

$$\begin{aligned} \underset{\nu}{\text{minimize}} \quad & \|\nu\|_{L^0} + \alpha\|x(T) - x_d\| \\ \text{subject to} \quad & x(0) = x_0, \ \dot{x}(t) = A(t)x(t), \\ & A(t) = \mathscr{A}(t) + \bar{\mathscr{A}}(t), \\ & \bar{\mathscr{A}}(t) = \sum_{e=1}^{m} \nu_e(t) M_e(t), \\ & \nu(t) \in \{0, 1\}^m. \end{aligned}$$

◁

The main control goal of this problem is also to bring the end state $x(t)$ close to x_d like Problem 8.2, not by using conventional exogenous input signals, but by modifying the A-matrix in a time-varying manner. Specifically, if the nominal A-matrix is given by $\mathscr{A}(t)$ and $\nu_e(t) = 1$, $M_e(t)$ can be added to the A-matrix. The problem is a scheduling problem to find which modifications should be made at which moment to achieve the control goal with as few time modifications as possible.

8.3.1.2 Theoretical Results

In each of these problems, sparsity is expressed by the L^0 or ℓ^0 norms, which are not convex with respect to their argument. Indeed, these problems are essentially combinatorial optimization problems that are difficult to solve. In sparse optimization in signal processing, many efficient computational algorithms have been obtained by relaxing the 0-norm to the 1-norm [23]. However, the optimal solution obtained by the relaxation problem is often different from the original problem.

The problem discussed in the previous section seems to be even more difficult because the decision variables are infinite-dimensional (i.e., a combinatorial optimization problem with an infinite number of decision variables), but in fact, it has the opposite property: the convex relaxation problem gives an exact solution.[2]

Theorem 8.2 *Under certain moderate assumptions, the same optimal solution can be obtained by replacing* $\|\cdot\|_{L^0}$, $\|\cdot\|_{\ell^0}$ *with* $\|\cdot\|_{L^1}$, $\|\cdot\|_{\ell^1}$ *in Problem 8.2, respectively. Moreover, every element of the optimal solution takes one of the binary values* $\{0, 1\}$ *at all times.* ◁

A similar result is obtained when the decision variable is restricted to discrete values.

Theorem 8.3 *Under certain moderate assumptions, the same optimal solution can be obtained by replacing* $\|\cdot\|_{L^0}$ *with* $\|\cdot\|_{L^1}$ *and* $\{0, 1\}^m$ *with* $[0, 1]^m$ *in Problem 8.3, respectively.* ◁

The following result holds for Problem 8.4 [37].

[2] Similar results hold for input-affine non-linear stochastic systems [42].

Theorem 8.4 *Under certain mild assumptions, the same optimal solution can be obtained by replacing* $\|\cdot\|_{L^0}$ *with* $\|\cdot\|_{L^1}$ *and* $\{0, 1\}^m$ *with* $[0, 1]^m$ *in Problem 8.4, respectively.* $\triangleleft$

For both cases, the set of decision variables (functions on $[0, T]$ taking values in $[0, 1]^m$) is convex. Therefore, if time discretization is carried out with a sufficiently small time interval, the relaxation problems in Problems 8.2 and 8.3, in which the evaluation function is linear with respect to the decision variables, can be solved using standard convex optimization solvers such as CVX. The relaxation problem in Problem 8.4 can also be solved using non-linear optimization solvers such as `fmincon` (MATLAB).[3]

8.3.2 Application to the Rebalancing Problem

In one-way car sharing services, users are free to choose where to return their vehicles, which may result in an unbalanced number of vehicles at each station. Therefore, the smooth operation of the service requires the adjustment (rebalancing) of the number of vehicles by the service provider at each station. Recently, the optimization problem for this vehicle transport scheduling has been studied [6, 77]. Here, we formulate the rebalancing problem as a sparse optimal control problem.

8.3.2.1 Model of Car-Sharing Service

In this section, we introduce the dynamics involved in rebalancing [40]. If the mathematical depth presented here is not of interest, the reader can skip the rebalancing dynamics described below and progress on to the next section. A total of s stations are labeled by $\mathcal{S} = \{1, 2, \ldots, s\}$ and the amount of vehicle usage per unit time from station $i \in \mathcal{S}$ to $j \in \mathcal{S}$ is denoted by $g_{ij}(t) \geq 0$. Similarly, let $u_{ij}(t) \geq 0$ denote the amount of rebalancing per unit time from station i to j by the staff ($g_{ii}(t) = u_{ii}(t) = 0, \forall i, t$). First, the number of vehicles at station i is denoted by $v_i(t) \in \mathbb{R}$ and its dynamics is assumed to follow:

$$\dot{v}_i(t) = \sum_{j=1}^{s} \gamma_{ij} f_{ij}(t) - \sum_{j=1}^{s} (g_{ij}(t) + u_{ij}(t)), \tag{8.24}$$

[3] In [37], a numerical method for solving a problem similar to Problem 8.4 with Newton Raphson method is proposed.

where $\gamma_{ij} = \frac{1}{\tau_{ij}}$ holds, and $\tau_{ij} = \tau_{ji} > 0$ represents the amount of time required for a user to travel between stations i and j, which is generally determined by the distance between i, j, and traffic conditions. Furthermore, the dynamics of the number of vehicles f_{ij} that are driven by users from station $j \in \mathcal{S}$ to $i \in \mathcal{S}$ is assumed to be

$$\dot{f}_{ij}(t) = -\gamma_{ij} f_{ij}(t) + g_{ij}(t) + u_{ij}(t). \tag{8.25}$$

Next, we introduce the concept of dynamic pricing. The usage fee from station j to i is denoted by $p_{ij}(t) \in \mathbb{R}$. Based on the model in the literature [61], the amount of movement between each station is assumed to vary with price, as follows:

$$g_{ij}(t) = \bar{g}_{ij}(t) - \theta_{ij}(t) p_{ij}(t). \tag{8.26}$$

However, $\bar{g}_{ij}(t) \in \mathbb{R}$ is the expected number of vehicles when $p_{ij}(t) = 0$, and $\theta_{ij}(t) = \theta_{ji}(t) \geq 0$ denotes price elasticity. Moreover, by using standard price

$$\bar{p}_{ij}(t) = \frac{\bar{g}_{ij}(t)}{\theta_{ij}(t)}, \tag{8.27}$$

assume that the price $p_{ij}(t)$ is determined according to the number of vehicles at stations i and j by the following equation:

$$p_{ij}(t) = \bar{p}_{ij}(t) + \nu_{ij}(t) v_i(t) - \nu_{ji}(t) v_j(t). \tag{8.28}$$

However, $\nu_{ij}(t), \nu_{ji}(t) \in \{0, 1\}$ are parameters that determine whether the price p_{ij} is adjusted according to the number of vehicles at stations i and j. Under such formulation, the price $p_{ij}(t)$ increases as the number of vehicles $v_i(t)$ increases, with the effect of reducing the number of vehicles to station i (and vice versa).

Therefore, by summarizing the above arguments, the following dynamics are obtained:

$$\dot{v}_i(t) = \sum_{j=1}^{s} (\gamma_{ij} f_{ij}(t) + \theta_{ji}(t)(\nu_{ji}(t) v_j(t) - \nu_{ij}(t) v_i(t)) - u_{ji}(t)), \tag{8.29}$$

$$\dot{f}_{ij}(t) = -\gamma_{ij} f_{ij}(t) - \theta_{ij}(t)(\nu_{ij}(t) v_i(t) - \nu_{ji}(t) v_j(t)) + u_{ij}(t). \tag{8.30}$$

Finally, by defining

$$
\begin{aligned}
&x(t) = [v_1(t)\ \cdots\ v_s(t)\ f_{12}(t)\ f_{13}(t) \cdots f_{s,s-1}(t)]^\top,\\
&u(t) = [u_{12}(t)\ u_{13}(t)\ \cdots\ u_{s,s-1}(t)]^\top,\\
&\mathcal{A} = \begin{bmatrix} 0 & \Gamma \\ 0 & -\Delta \end{bmatrix},\ \bar{\mathcal{A}}(t) = \begin{bmatrix} \Xi\Lambda(t) & 0 \\ -\Lambda(t) & 0 \end{bmatrix},\ B(t) = \begin{bmatrix} -\Xi \\ I \end{bmatrix},\\
&\Xi = [\hat{E}_1 \hat{E}_2\ \cdots \hat{E}_s]^\top,\ \hat{E}_i = [e_1 \cdots e_{i-1} e_{i+1}\ \cdots\ e_s]^\top,\\
&\Lambda(t) = \begin{bmatrix} \theta_{12}(\nu_{12}(t)e_1 - \nu_{21}(t)e_2)^\top \\ \theta_{13}(\nu_{13}(t)e_1 - \nu_{31}(t)e_3)^\top \\ \vdots \\ \theta_{s,s-1}(\nu_{s,s-1}(t)e_s - \nu_{s-1,s}(t)e_{s-1})^\top \end{bmatrix},\\
&\Gamma = [e_1\gamma_{12}\ e_1\gamma_{13}\ \cdots e_s\gamma_{s,s-1}]^\top,\\
&\Delta = \mathrm{diag}(\gamma_{12}\ \gamma_{13}\ \cdots \gamma_{s,s-1})
\end{aligned}
$$

with $n = s^2$, $m = s^2 - s$ and $e_i \in \mathbb{R}^n$ as ith standard basis, the dynamics of the whole system can be summarized as follows:

$$\dot{x}(t) = (\mathcal{A} + \bar{\mathcal{A}}(t))x(t) + B(t)u(t). \tag{8.31}$$

- $x(t)$ is the number of waiting cars at each station
- $u(t)$ is the amount of rebalancing carried out between each station
- $\nu(t)$ is whether or not dynamic splines are implemented between each station
- $\bar{\mathcal{A}}(t)$ is linear with respect to each element of $\nu(t)$.

8.3.2.2 Sparse Rebalancing Problem

Here, the rebalancing problem in car-sharing systems is formulated as a sparse optimal control problem.

Problem 8.2' The initial number of vehicles $v_0 \in \mathbb{R}^s$ at all stations is generally nonuniform and needs to be adjusted by the staff to the required number of vehicles $v_d \in \mathbb{R}^s$ to provide a smooth service. Consequently, regarding the state variable $x(t)$ in dynamics (8.31), we consider the problem of designing an input $u(t)$ for transitions from the initial state $x(0) = x_0$ to the target state $x(T) = x_d$ within T. Here, each $u_{ij}(t)$ corresponds to the number of vehicles that are rebalanced by staff between stations i and j. Therefore, when the number of staff teams is β, the number of vehicles that can be transported simultaneously is at most β, and the constraint $\|u(t)\|_{\ell^0} \le \beta$ is imposed on the input variables. In addition, as each team can only transport one vehicle at a time, $u(t) \in [0, 1]$ must be satisfied. Furthermore, as it is desirable for staff to have a shorter transportation time, $\|u\|_{L^0}$ is defined as an evaluation function to achieve this goal. In other words, in the rebalancing scheduling, considering the cost of staff and treating this as a sparse optimal control problem, it is formulated as in Problem 8.2.

Problem 8.3' Problem 8.2' is a problem of determining the specific rebalancing implementation method with given x_d. In contrast, although x_d is still undetermined in the previous day's planning, having staff standing by at all stations all the time is not cost-efficient. Therefore, staff scheduling has to be determined to achieve effective rebalancing for any x_d. As the minimum energy required (on average) when the target state is unknown is characterized by the controllability Gramian, the decision of which station to keep staff ready and when can be formulated as in Problem 8.3.

Problem 8.4' Next, we formulate an optimal control problem for rebalancing to reduce the time spent on dynamic pricing by the service provider. Even if dynamic pricing is carried out over a long period of time, there may be situations in which users become *used to it*, and it does not have the expected effect. We assume that the service provider decides whether or not to implement $\nu_{ij}(t)$. This means that dynamic pricing scheduling, which determines at which time and between which stations price interventions are made, can be formulated as in Problem 8.4.

8.3.2.3 Numerical Examples

To demonstrate the usefulness of the proposed method, numerical calculations are performed after convex relaxation for Problems 8.2' and 8.4'. The number of stations was set to $s = 10,\ T = 4$, the initial value of user vehicles at each station was determined according to an independent uniform distribution of $[10, 30]$, and the target state x_d was set to the average of the total number of user vehicles. Parameters such as the arrival rate γ_{ij} and the coefficient of price elasticity $\theta_{ij}(t)$ were determined in the same way as in [40].

First, the optimal trajectory u_{ij} resulting from solving Problem 8.2' under

$$\nu_{ij}(t) = 1,\ \forall i,\ j,\ t, \tag{8.32}$$

assuming that dynamic pricing is effective at all times and between stations, is shown in Fig. 8.7. This confirms that the optimal solution takes discrete values despite solving a convex relaxation problem. Note that there are times when values other than $\{0, 1\}$ are taken near the changeover of each trajectory, but this is due to the effect of time discretization. In other words, the time width must be sufficiently small to prevent such phenomena from becoming problematic. The total time length (L^0 norm of u_{ij}) with rebalancing between stations is shown in Fig. 8.8. It can be seen that rebalancing is not performed between most stations (blue), which confirms the sparsity of the optimal solution.

Consider the situation in which rebalancing by staff transport is not regarded here (i.e., $u = 0$), and the time to implement dynamic pricing is suppressed. The terminal error $\|x(T) - x_d\|$ and the total time length of dynamic pricing $\|\nu\|_{L^0}$ are shown in

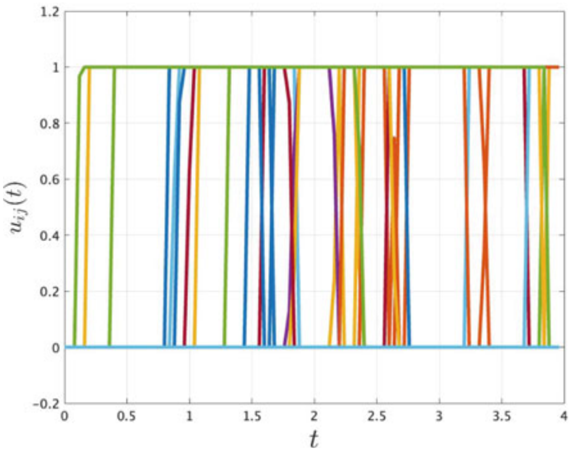


Fig. 8.7 Each trajectory u_{ij} in the optimal solution of Problem 8.2'

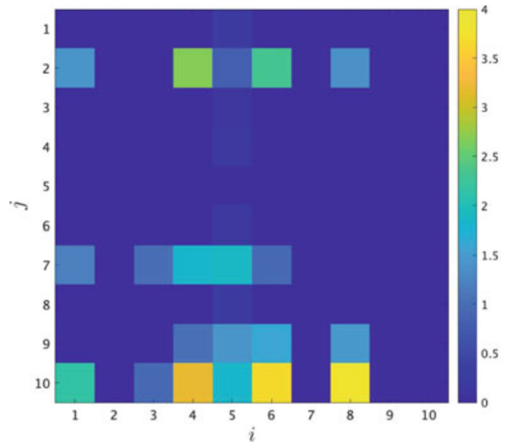


Fig. 8.8 Total time $\|u_{ij}\|_{L^0}$ of rebalancing implementation between each station in the optimal solution of Problem 8.2'

Fig. 8.9 when the optimal ν for each parameter α is used. Both are shown as a ratio to the case of (8.32), and the number on each point is the value of α. Naturally, there is a trade-off between these two values; however, it can be seen, for example, that the ratio of $\|\nu\|_{L^0}$ can be kept below 0.1, while the ratio of the terminal error can be reduced to approximately 1.3.

Finally, to confirm the implementation of dynamic pricing switches each time, snapshots of the optimal ν for $\alpha = 100$ at $t = 16h, 24h, 32h$ are shown in Fig. 8.10, where h is the discretized step size. Yellow and blue in Fig. 8.10 represent $\nu_{ij}(t) = 1$ and $\nu_{ij}(t) = 0$, respectively, and it can be seen that the implementation point changes with time.

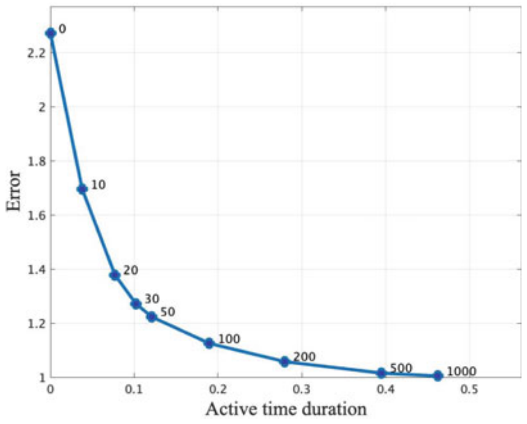


Fig. 8.9 Implementation time length $\|\nu\|_{L^0}$ and terminal error $\|x(T) - x_d\|$ (ratio to the fully active time) in the optimal solution of Problem 8.4'

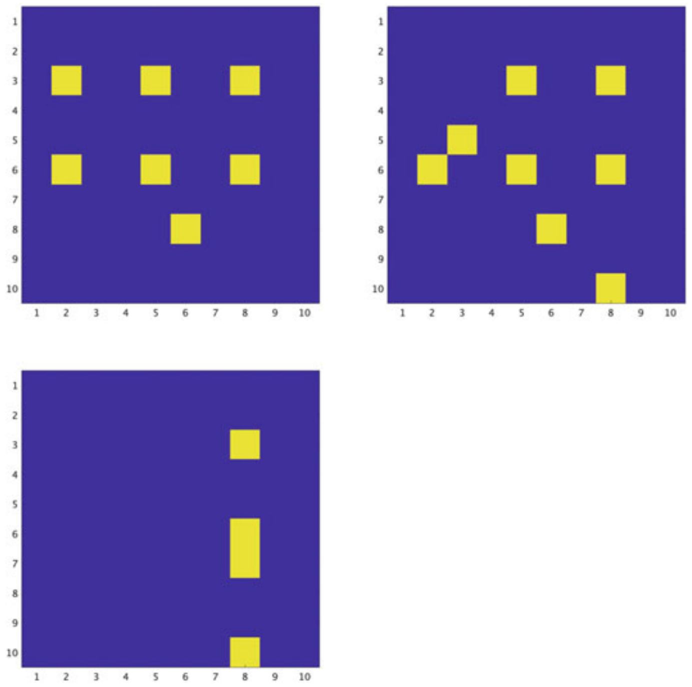


Fig. 8.10 Snapshots of the price adjustment implementation point ν_{ij} in the optimal solution of Problem 8.4' (Top: $t = 16h$, Middle: $t = 24h$, Bottom: $t = 32h$)

8.4 Communication-Aware Cooperative Car-Sharing Control

In this section, we present a framework of cooperative rebalancing control of one-way car-sharing service based on distributed control and optimization [27, 28, 43, 57, 58, 76]. We consider a group of service providers operate vehicles independently but share common rental stations. Since the deadheading of vehicles involves extra costs for the service providers, the deadhead vehicles should be minimized as much as possible. The cooperative rebalancing control to minimize the deadhead vehicles is formulated as a distributed optimization problem with a coupling inequality. The optimization problem can be solved by consensus-based algorithms [10, 11, 17, 79]. However, the conventional distributed optimization methods require communication between service providers at each iteration of the optimization algorithm, which may result in waste of communication resources [29]. To save limited communication resources, the distributed event-triggered methods have been considered in many applications of control and optimization [1, 7, 8, 25, 26, 46, 50]. In this section, we review a distributed primal-dual method with event-triggered communication [24], in which service providers communicate with other providers when the error of the estimated solution exceeds a predefined threshold. Each service provider updates the estimations of the primal and dual optimizers using a consensus-based primal-dual algorithm. We confirm that the estimations of all providers converge to an optimal rebalancing solution under the constraints of available vehicles and parking slots.

8.4.1 Formulation of Cooperative Rebalancing Control

We consider N (≥ 1) service providers that cooperatively operate the car-sharing service among S (≥ 2) stations.

Let $a_{i,j}[k] \in \mathbb{R}_+$ and $f_{i,j\ell}[k] \in \mathbb{R}_+$ be the expected number of vehicles of provider $i \in \mathcal{V}$ parked at station $j \in \mathcal{S}$ at time $k \in \mathbb{N}$ and the expected number of vehicles of provider $i \in \mathcal{V}$ traveling from the station $\ell \in \mathcal{S}$ to $j \in \mathcal{S}$ at time $k \in \mathbb{N}$, where $\mathcal{V} = \{1, 2, \ldots, N\}$ is the set of the service providers and $\mathcal{S} = \{1, 2, \ldots, S\}$ is the set of the stations. Suppose also that $s_{i,j\ell}[k] \in \mathbb{R}_+$ and $w_{i,j\ell}[k] \in \mathbb{R}_+$ are respectively the expected rates of the number of deadhead vehicles and the demands of provider i from station ℓ to station j at time k. Then, the dynamics of the car-sharing system of provider i is given as follows:

$$a_{i,j}[k+1] = a_{i,j}[k] + \sum_{\ell \in \mathcal{S} \setminus \{j\}} c_{j\ell}[k] f_{i,j\ell}[k] - \sum_{\ell \in \mathcal{S} \setminus \{j\}} (s_{i,\ell j}[k] + w_{i,\ell j}[k]), \quad j \in \mathcal{S}, \tag{8.33}$$

$$f_{i,j\ell}[k+1] = (1 - c_{j\ell}[k]) f_{i,j\ell}[k] + s_{i,j\ell}[k] + w_{i,j\ell}[k], \quad j, \ell \in \mathcal{S},\ j \neq \ell, \tag{8.34}$$

where $c_{j\ell}[k] = 1 - e^{-1/\tau_{j\ell}[k]}$ is the expected arrival rate from station ℓ to station j.

The dynamics (8.33) shows that the number of the operator i's vehicles at station j for the subsequent time $k+1$, denoted as $a_{i,j}[k+1]$, is determined by the sum of the current number of vehicles at station j and the influx of vehicles from other stations, subtracted by the number of vehicles departing from station j toward other stations. Similarly, the dynamics (8.34) shows that the number of the operator i's vehicles in transit from station ℓ to j at the next time $k+1$, represented as $f_{i,j\ell}[k+1]$, equals the number of vehicles currently on route from station ℓ to j that won't reach j by the next time, added to the total vehicles initiating their journey from station ℓ toward j.

8.4.2 Distributed Event-Triggered Optimization

We consider the case when several service providers jointly use common car stations. Because the parking space of the stations is shared among providers, each provider manages the deadhead vehicles so that the number of these vehicles is as small as possible under the constraints of the parking space.

Let $x_i[k] = [a_{i,1}[k], \ldots, a_{i,S}[k], f_{i,12}[k], \ldots, f_{i,S,S-1}[k]]^\top \in \mathbb{R}^n$, $u_i[k] = [s_{i,21}[k], s_{i,31}[k], \ldots, s_{i,S-1,S}[k]]^\top \in \mathbb{R}^m$, and $w_i[k] = [w_{i,21}[k], w_{i,31}[k], \ldots, w_{i,S-1,S}[k]]^\top \in \mathbb{R}^m$ be the state, the control input, and the external input of provider i at time k, where $m = S(S-1)$ and $n = S^2$. From (8.33) and (8.34), the dynamics of the car-sharing service is given as

$$x_i[k+1] = A_i x_i[k] + B_i(u_i[k] + w_i[k]), \quad x_i[0] = x_{i0}, \tag{8.35}$$

where $A_i \in \mathbb{R}^{n \times n}$ and $B_i \in \mathbb{R}^{n \times m}$ are the coefficient matrices, and x_{i0} ($\geq 0_n$) is the initial state of provider i.

Let $u_i = [u_i^\top[0],\ u_i^\top[1],\ \ldots,\ u_i^\top[T-1]]^\top \in \mathbb{R}^{mT}$ be the vector that represents the number of deadhead vehicles of provider i during the time horizon $[0, T-1]$. Because both the expected number of vehicles between stations and the parked vehicles are non-negative, the non-negativeness condition for the state is required as follows:

$$x_i[k] \geq 0_n, \quad \forall i \in \mathcal{V},\ \forall k \in \mathcal{K}, \tag{8.36}$$

where $\mathcal{K} = \{1, 2, \ldots, T\}$ and the inequality sign for vectors represents the element-wise operation. The constraint on the limited parking space is represented as

$$\Gamma \sum_{i=1}^{N} x_i[k] \leq \xi[k], \quad \forall k \in \mathcal{K}, \tag{8.37}$$

where $\Gamma = [I_S \ \ O_{S\times(n-S)}] \in \mathbb{R}^{S\times n}$ extracts the variables related to the number of vehicles at stations and $\xi[k] \in \mathbb{R}^S$ represents the number of vehicles that can be parked at S stations at time k. The constraint on the number of deadhead vehicles is given as

$$u_i \in \mathcal{U}_i = \{u' \in \mathbb{R}^{mT} \mid 0_{mT} \leq u' \leq \eta_i\}, \quad \forall i \in \mathcal{V}, \tag{8.38}$$

where $\eta_i \in \mathbb{R}^{mT}$ is the upper bound of the number of dispatchable vehicles of provider i.

Then, the cooperative rebalancing control problem is summarized as follows.

Problem 8.5 For the dynamics of the car-sharing service (8.33) and (8.34), find the control inputs $\{u_i\}_{i=1}^N$ such that the number of deadhead vehicles is as small as possible under the constraints of dispatchable vehicles and parking space of the stations (8.36)–(8.38).

The distributed optimization problem is formulated as follows:

$$\underset{\{u_i\}_{i=1}^N}{\text{minimize}} \quad \sum_{i=1}^{N} f_i(u_i) \tag{8.39a}$$

$$\text{subject to} \quad \sum_{i=1}^{N} g_i(u_i) \leq 0_q, \tag{8.39b}$$

$$u_i \in \mathcal{U}_i, \quad i \in \mathcal{V}. \tag{8.39c}$$

In the distributed optimization problem (8.39), $f_i(u_i) = u_i^\top H_i u_i$, where $H_i = \sum_{\tau=0}^{T-1} E^\top[\tau] R_i E[\tau] \in \mathbb{R}^{mT\times mT}$, $E[k] = [O_{m\times mk}, \ \ I_m, \ \ O_{m\times m(T-k-1)}] \in \mathbb{R}^{m\times mT}$, and $R_i \in \mathbb{R}^{m\times m}$ is a positive-definite weight matrix. Moreover, $g_i(u_i) = \Omega_i u_i + \psi_i$, where $\Omega_i = \left[-\hat{\Phi}_i^\top, \check{\Phi}_i^\top\right]^\top \in \mathbb{R}^{q\times mT}$, $\psi_i = \left[-(\hat{A}_i \xi_i[0])^\top, \mu_i^\top\right]^\top \in \mathbb{R}^q$, and $q = (n + S)T$. The matrices $\hat{\Phi}_i$, $\check{\Phi}_i$, and $\hat{A}_i$, and the vector μ_i are given by $\hat{\Phi}_i = [\Phi_i^\top[1], \Phi_i^\top[2], \ldots, \Phi_i^\top[T]]^\top \in \mathbb{R}^{nT\times mT}$, $\check{\Phi}_i = [(\Gamma\Phi_i[1])^\top, (\Gamma\Phi_i[2])^\top, \ldots, (\Gamma\Phi_i[T])^\top]^\top \in \mathbb{R}^{ST\times mT}$, $\hat{A}_i = [A_i^\top, (A_i^2)^\top, \ldots, (A_i^T)^\top]^\top \in \mathbb{R}^{nT\times n}$, and

$$\mu_i = \begin{bmatrix} \Gamma \sum_{i=1}^N A_i x_i[0] + \Gamma \sum_{i=1}^N \Phi_i[1] w_i - \frac{1}{N}\xi[1] \\ \Gamma \sum_{i=1}^N A_i^2 x_i[0] + \Gamma \sum_{i=1}^N \Phi_i[2] w_i - \frac{1}{N}\xi[2] \\ \vdots \\ \Gamma \sum_{i=1}^N A_i^T x_i[0] + \Gamma \sum_{i=1}^N \Phi_i[T] w_i - \frac{1}{N}\xi[T] \end{bmatrix} \in \mathbb{R}^{ST}.$$

The dual problem for (8.39) is given as follows:

$$\begin{aligned} &\text{maximize} \quad \sum_{i=1}^N \varphi_i(\lambda) \\ &\text{subject to} \quad \lambda \geq 0_q, \end{aligned} \tag{8.40}$$

where $\varphi_i(\lambda) = \min_{u_i \in \mathcal{U}_i} \mathcal{L}_i(u_i, \lambda)$ and $\mathcal{L}_i(u_i, \lambda) = f_i(u_i) + \lambda^\top g_i(u_i)$ is the Lagrange function of provider i. The Lagrange function for (8.39) is given as $\mathcal{L}(u, \lambda) = \sum_{i=1}^N \mathcal{L}_i(u_i, \lambda)$, where $u = [u_1^\top, u_2^\top, \ldots, u_N^\top]^\top \in \mathcal{U} \subset \mathbb{R}^{mNT}$.

Let $f(u) = \sum_{i=1}^N f_i(u_i)$ and $g(u) = \sum_{i=1}^N g_i(u_i)$ be the global objective function and the global constraint function. We assume Slater's qualification for (8.39) [79].

Assumption 8.1 There exists $u \in {}^\circ\mathcal{U}$ such that $g(u) < 0_q$, where ${}^\circ\mathcal{U}$ is the relative interior of $\mathcal{U}$.

If Slater's qualification holds, we have $f^* = \varphi^*$ and $\Lambda^* \neq \emptyset$ [79], where φ^* is the optimal value of (8.40) and Λ^* is the set of the optimal dual solutions of (8.40).

To solve the optimization problem (8.39), each service provider exchanges the estimation of the optimal solution of the dual problem (8.40) with other providers via a communication network. In this study, the communication network between providers is represented by an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{E} \subset \{\{i, j\} \mid i, j \in \mathcal{V},\ i \neq j\}$ is the set of undirected edges. The set of neighbors of provider i is denoted as $\mathcal{N}_i = \{j \in \mathcal{V} \mid \{i, j\} \in \mathcal{E}\}$. We make the following assumption for the communication network.

Assumption 8.2 The communication network $\mathcal{G}$ is connected.

8.4.3 *Event-Triggered Distributed Primal-Dual Algorithm*

Let $u_i^* \in \mathcal{U}_i^*$ and $\lambda^* \in \Lambda^*$ be an optimal primal solution for (8.39) and an optimal dual solution for the dual problem (8.40), where $\mathcal{U}_i^* \subset \mathcal{U}_i$ is the set of the optimal primal solutions of provider i. Each provider i has the estimations of the optimal primal solution $u_i^{(s)} \in \mathbb{R}^{mT}$ and the optimal dual solution $\lambda_i^{(s)} \in \mathbb{R}^q$ at iteration $s \in \mathbb{N}$ of the distributed algorithm.

Each service provider sends the estimation of the optimal dual solution $\lambda_i^{(s)}$ to the neighboring providers only when the difference between the current and most recently transmitted estimates exceeds a certain threshold. The iteration in which service provider i sends the estimation to the neighboring providers is called the trigger time of provider i. Let s_i^ℓ be the ℓth trigger time of provider i, where $\ell \in \mathbb{N}$. The latest estimation sent by provider i to the neighboring providers is defined as follows:

$$\tilde{\lambda}_i^{(s)} = \begin{cases} \lambda_i^{(s)}, & \text{if } s \in \kappa_i, \\ \tilde{\lambda}_i^{(s-1)}, & \text{if } s \notin \kappa_i, \end{cases} \tag{8.41}$$

where $\kappa_i = \{s_i^0, s_i^1, \dots\}$ is the set of the trigger times of provider i. In the event-triggered algorithm, each service provider i sends the estimation to neighboring providers only at iterations that satisfy the following condition:

$$\|\lambda_i^{(s)} - \tilde{\lambda}_i^{(s)}\| \geq \varepsilon_i^{(s)}, \tag{8.42}$$

where $\varepsilon_i^{(s)} > 0$ is a trigger threshold of provider i. Then, for any $i \in \mathcal{V}$, we get

$$\|\lambda_i^{(s)} - \tilde{\lambda}_i^{(s)}\| \leq \varepsilon^{(s)}, \quad \forall s \in \mathbb{N}, \tag{8.43}$$

where $\varepsilon^{(s)} = \max_{i \in \mathcal{V}} \varepsilon_i^{(s)}$.

Assumption 8.3 The upper bound of the allowable error satisfies

$$\lim_{s\to\infty} \varepsilon^{(s)} = 0, \quad \sum_{s=0}^{\infty} (\varepsilon^{(s)})^2 < \infty, \quad \sum_{s=0}^{\infty} \varepsilon^{(s)} = \infty.$$

We define Λ_i as $\Lambda_i = \{\lambda \in \mathbb{R}^q_{\geq 0} \mid \|\lambda\| \leq (f(\check{u}) - \check{\varphi})/\Delta(\check{u}) + \omega_i\}$ for some $\omega_i > 0$, where $\check{u} \in \mathcal{U}$ is a point that satisfies Assumption 8.1, $\Delta(u) = \min_{1\leq p\leq q} \{-[\sum_{i=1}^N g_i(u_i)]_p\}$, and $\check{\varphi} = \min_{u\in\mathcal{U}} \mathcal{L}(u, \lambda)$ for $\lambda \in \mathbb{R}^q_+$.

After the information exchange between the neighboring providers, provider i updates the estimations of the optimal solutions of the primal problem (8.39) and the dual problem (8.40) by the following event-triggered distributed primal-dual algorithm:

$$w_i^{(s)} = \lambda_i^{(s)} + \sum_{j=1}^{N} p_{ij} \left(\tilde{\lambda}_j^{(s)} - \tilde{\lambda}_i^{(s)}\right), \tag{8.44}$$

$$\lambda_i^{(s+1)} = \Pi_{\Lambda_i} \left[w_i^{(s)} + \alpha^{(s)} d\mathcal{L}_{\lambda,i}(u_i^{(s)}, w_i^{(s)})\right], \tag{8.45}$$

$$u_i^{(s+1)} = \Pi_{\mathcal{U}_i} \left[u_i^{(s)} - \alpha^{(s)} d\mathcal{L}_{u,i}(u_i^{(s)}, w_i^{(s)})\right], \tag{8.46}$$

where $d\mathcal{L}_{\lambda,i}$ and $d\mathcal{L}_{u,i}$ are the subgradients of $\mathcal{L}_i$ with respect to λ and u_i, and $\Pi_C[x]$ is the projection of a point $x \in \mathbb{R}^n$ to a closed convex set C.

Assumption 8.4 The step-size satisfies

$$\lim_{s\to\infty} \alpha^{(s)} = 0, \quad \sum_{s=0}^{\infty} (\alpha^{(s)})^2 < \infty, \quad \sum_{s=0}^{\infty} \alpha^{(s)} = \infty.$$

Assumption 8.5 The weight p_{ij} is given as

$$p_{ij} = \begin{cases} p_{ji} > p_{\min}, & \text{if } \{i, j\} \in \mathcal{E}, \\ 0, & \text{if } \{i, j\} \notin \mathcal{E},\ i \neq j, \end{cases}$$
$$p_{ii} = 1 - \sum_{j \in \mathcal{V}\setminus\{i\}} p_{ij} > p_{\min},$$

where $0 < p_{\min} < 1$.

In the event-triggered distributed primal-dual algorithm, the information on the estimation $u_i^{(s)}$, the local cost function f_i, and the local constraints of sharing services (8.36)–(8.38) are not disclosed to the other providers. Therefore, any service provider can determine the appropriate control input while retaining the anonymity of their customers' information.

The following theorem shows that the estimations of all providers converge to the optimal solutions of (8.39) and (8.40) [24].

Theorem 8.5 *If each provider updates the estimations by the distributed event-triggered primal-dual algorithm* (8.44)–(8.46)*, then* $\lim_{s\to\infty} \|u_i^{(s)} - u_i^*\| = 0$ *and* $\lim_{s\to\infty} \|\lambda_i^{(s)} - \lambda^*\| = 0$ *for any* $i \in \mathcal{V}$.

Theorem 8.5 shows that each provider can cooperatively solve Problem 8.5 by the proposed event-triggered primal-dual algorithm.

8.4.4 Numerical Simulation

We consider a car-sharing problem with $N = 4$ and $S = 15$, in which each provider dispatches up to 400 vehicles for the car-sharing service. The maximum number of vehicles that can be parked at each time is given by $\xi_j[k] = 50$ for any station j. The expected demands of each provider were given randomly from the interval between 0 and 15. The arrival rate $c_{j\ell}$ is given in the interval [0.95, 0.99] and the maximum number of dispatchable vehicles η_i is 10. The step-size is given as $\alpha^{(s)} = 1/(s + 2 \times 10^7)^{0.51}$ and the trigger function for the event-triggered algorithm is given as $\varepsilon_i^{(s)} = C_E/(s + 1)$ for all $i \in \mathcal{V}$, where C_E is a positive constant. The weight matrix is set as $R_i = I_m$ for $i \in \mathcal{V}$.

Figures 8.11 and 8.12 show the normalized error $(\sum_{i=1}^{n} |f(u_i^{(s)}) - f^*|)/(\sum_{i=1}^{n} |f(u_i^{(0)}) - f^*|)$, and the number of triggers for the different values of the threshold parameter C_E of the event-triggered communication, where f^* is the cost

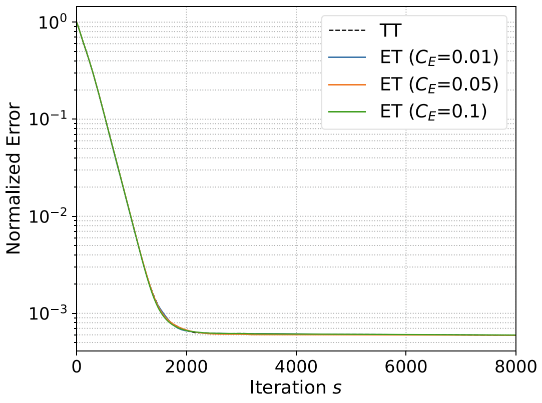


Fig. 8.11 The normalized errors with different values of C_E

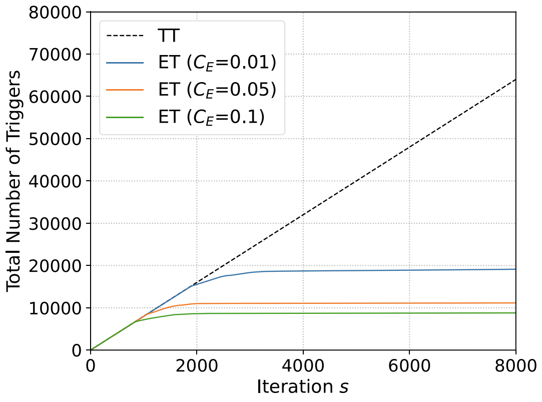


Fig. 8.12 The number of triggers with different values of C_E

obtained by a centralized computation with CVXPY [16]. These results show that the accuracy of the estimation by the event-triggered algorithm (ET) is comparable to that by the time-triggered algorithm (TT) [79]. Moreover, the number of communications by the event-triggered algorithm can be efficiently reduced compared with that by the time-triggered algorithm.

8.5 Optimal Control Problem of Probability Distributions for Rebalancing in One-Way Car-Sharing Service

This section treats the operation of car-sharing services as a control problem of probability distributions, which is a current focus in control theory [2, 12, 22, 32]. As discussed in previous sections, we examine the rebalancing problem in one-way car-sharing services, where users can return vehicles to arbitrary slots, not necessarily the ones from which the users rented them. This can result in an uneven distribution of vehicles at parking slots. To address this issue, the service operator performs a rebalancing operation by dispatching the vehicles to eliminate the uneven distribution [5, 44, 48, 69]. The section assumes that car-sharing systems dispatch vehicles to match user demand, formulating the rebalancing problem as controlling the numbers of vehicles to match the demand. As discussed in the following, the number of vehicles at each station corresponds to the probability distribution, considering that the total number of vehicles remains the same (i.e., no additional cars are brought into or removed from the total pool), and so does the required number of vehicles in demand. Therefore, the rebalancing problem can be considered as the minimization of the difference between the probability distributions of the actual availability of vehicles and the demand for required vehicles. The section presents an approach based on the Wasserstein distance to measure this difference and formulates the rebalancing problem as the optimal control problem of probability distributions with Wasserstein costs.

8.5.1 Optimal Control Problem of Probability Distributions: Problem Formulation and Optimal Condition

Here we describe the optimal control problem of probability distributions with costs determined by the Wasserstein distances. The formulation of the control problem is based on [33] focusing on the discrete setting. The continuous setting can be found in [32].

The control problem focuses on discrete state variables, i.e., the state space consists of a finite set $S = \{s_1, s_2, \dots, s_N\}$. The problem takes probability distributions on the state space S into consideration. The value of the probability $\mathrm{P}[x_t = s_i]$ that an S-valued random variable x_t at time t $(t = 1, 2, \dots)$ takes the value $s_i \in S$ is denoted by

$$\mathrm{P}[x_t = s_i] = (p_t)_i, \tag{8.47}$$

where $(p_t)_i \in [0, 1]$ for $i \in \mathbb{Z}_{1:N}$. The values $(p_t)_i$ $(i = 1, \dots, N)$ satisfy $\sum_{i=1}^{N} (p_t)_i = 1$ for each t. The probability distribution of x_t is given by a probability vector $p_t = [(p_t)_1, (p_t)_2, \dots, (p_t)_N]^T$, consisting of the set of probability distributions $\mathcal{P}(S)$.

The above setting enables the introduction of Markov decision processes. We assume that the transition matrix of the Markov decision process depending on the control input $u_t \in U \subset \mathbb{R}^d$, for $t \in \mathbb{Z}_{0:T-1}$, is given in the form of

$$\mathrm{P}\left[x_{t+1} = s_i \mid x_t = s_j\right] = P_{ij}(u_t, t) \quad \text{for } \forall i, j \in \mathbb{Z}_{1:N}, \tag{8.48}$$

where $\mathrm{P}\left[x_{t+1} = s_i \mid x_t = s_j\right]$ is the conditional probability of the event $x_{t+1} = s_i$ under the event $x_t = s_j$. The elements of the matrix $P_{ij}(u_t, t)$ is a map from $U \times \mathbb{Z}_{0:T-1}$ to $[0, 1]$ and satisfies

$$\sum_{i=1}^{N} P_{ij}(u,t) = 1 \text{ for } u \in U,\ t \in \mathbb{Z}_{0:T-1}. \tag{8.49}$$

Then, a Markov chain is determined by

$$p_{t+1} = P(u_t,t)p_t, \quad t \in \mathbb{Z}_{0:T-1} \tag{8.50}$$

and the initial value of $p_0 \in \mathcal{P}(S)$. In this section, we discuss the application of the Markov decision process to the rebalancing in car-sharing services. As mentioned above and discussed in detail below, rebalancing through vehicle dispatching can be regarded as controlling the number of available vehicles to match user demand. We regard this matching problem as minimizing the difference between the number of vehicles and user demand. In terms of the Markov decision process, this problem reduces to the minimization between the actual and desired probability distributions of vehicles at parking slots when the numbers of vehicles at the parking slots are normalized.

To measure the difference between probability distributions in the rebalancing problem, the study [33] introduced Wasserstein distances. To introduce it, the state $s_i \in S$ is associated with a point located at z_i in the Euclidean space $\mathbb{R}^n$.

Definition 8.1 Assume that the states s_i, s_j are associated with the positions $z_i, z_j \in \mathbb{R}^n$, respectively, and assume that the distance between z_i and z_j is given by $d_{ij} = \|z_i - z_j\|$ $(i, j \in \mathbb{Z}_{1:N})$. For two probability distributions $p, q \in \mathcal{P}(S)$, consider the set of joint probability distributions π of S-valued random variables X and Y whose probability distributions p and q, respectively, are as follows, i.e., $\pi = \{\pi_{ij}\}$ satisfies

$$\pi_{ij} = \mathrm{P}\left[X = s_i,\ Y = s_j\right] \tag{8.51}$$

and

$$\sum_{j=1}^{N} \pi_{ij} = p_i, \quad \sum_{i=1}^{N} \pi_{ij} = q_j. \tag{8.52}$$

With the notation of the set of joint probability distributions π by $\Pi(p,q)$, the Wasserstein distance between p and q, $W : \mathcal{P}(S) \times \mathcal{P}(S) \to \mathbb{R}$, is given by

$$W(p,q) = \min_{\pi \in \Pi(p,q)} \left(\sum_{i,j=1}^{N} d_{ij}\pi_{ij} \right). \tag{8.53}$$

The set of joint probability distributions $\Pi(p,q)$ is called the coupling of probability distributions of p and q. Please note that the Wasserstein distance of Definition 8.1 is precisely referred to as the 1-Wasserstein distance because it uses the first power of the distance d_{ij}.

We use the following result to present an optimality condition of the control problem later. It follows from the definition that the Wasserstein distance is given as a solution of a linear programming problem. The following result gives the duality result of the linear programming problem.

Proposition 8.1 ([62]) *The Wasserstein distance admits another expression given by*

$$W(p, q) = \max_{(f,g)\in R(d)} f^T p + g^T q, \tag{8.54}$$

where the set $R(d)$ is defined as

$$R(d) = \left\{ f \in \mathbb{R}^N, g \in \mathbb{R}^N \mid \forall i, j \in \mathbb{Z}_{1:N},\ f_i + g_j \leq d_{ij} \right\}, \tag{8.55}$$

where f_i and g_j, $i, j \in \mathbb{Z}_{1:N}$ are the ith element of f and the jth element of g, respectively.

In expression (8.54), f and g are called Kantorovich potentials.

Given the Wasserstein distance, we can formulate the optimal control problem here. We consider the Markov decision process given by (8.50) and assume that the desired probability distributions $p_{t,d} \in \mathcal{P}(S)$ for each $t \in \mathbb{Z}_{1:T}$ are given. The problem formulated here is the minimization of the differences between the probability vector p_t of (8.50) and the desired distribution $p_{t,d}$ for each $t \in \mathbb{Z}_{1:T}$. As mentioned above, the differences between the two distributions are measured by the Wasserstein distance. The following gives the mathematical formulation of the control problem of this section.

Problem 8.6 Consider system (8.50), a probability distribution at the initial time, $p_0 \in \mathcal{P}(S)$ at $t = 0$, and desired probability distributions $p_{t,d}$, $t \in \mathbb{Z}_{1:T}$. Find a sequence of the control inputs $u = \{u_t\}_{t=0}^{T-1}$ that minimizes the cost function

$$J(u) = \sum_{t=0}^{T-1} \left\{ W(p_{t+1}, p_{t+1,d}) + g(p_t, u_t) \right\}, \tag{8.56}$$

where $g : \mathcal{P}(S) \times U \to \mathbb{R}$ is continuous on $\mathcal{P}(S) \times U$ and continuously differentiable on the interior of $\mathcal{P}(S) \times U$.

As seen below, we introduce the function $g(p_t, u_t)$ in (8.56) to impose the control cost. Additionally, $g(p_t, u_t)$ can also be used to take constraints on the control input such that $u_t \in U \subset \mathbb{R}^d$ into consideration, for example, using barrier functions for $u \in U$.

Problem 8.6 is the optimal control problem with respect to the probability distributions. In standard optimal control problems, Pontryagin's minimum principle enables the characterization of the optimality and the optimal control inputs. As for the current problem, we can obtain a condition based on Pontryagin's minimum principle.

Theorem 8.6 *Assume that given the initial probability distribution $p_0 \in \mathcal{P}(S)$ and the target probability distributions $p_{t,d} \in \mathcal{P}(S)$, $t \in \mathbb{Z}_{1:T}$, for the system (8.50), there exists a sequence of control inputs $\left\{u_t^*\right\}_{t=0}^{T-1}$ that attains the minimum of the cost function (8.56). Then, there exist sequences $\left\{p_t^*\right\}_{t=0}^{T}$ and $\left\{\Lambda_t^*\right\}_{t=0}^{T}$ such that the following conditions hold:*

$$p_{t+1}^* = P(u_t^*, t)p_t^*, \quad \Lambda_t^* = \Lambda_{t+1}^* P(u_t^*, t) + f_t^{*T} + \frac{\partial g}{\partial p}(p_t^*, u_t^*) \tag{8.57}$$

$$\Lambda_{t+1}^* \frac{\partial P}{\partial u_m}(u_t^*, t)p_t^* + \frac{\partial g}{\partial u_m}(p_t^*, u_t^*) = 0, \quad t \in \mathbb{Z}_{1:T-1}, \ m \in \mathbb{Z}_{1:d}, \tag{8.58}$$

with boundary conditions

$$p_0^* = p_0, \quad \Lambda_T^* = f_T^{*T}, \tag{8.59}$$

where f_t^*, $t \in \mathbb{Z}_{1:T}$ is a Kantorovich potential of $W(p_t^*, p_{t,d})$ as given in (8.54).

Note that Λ_t^* is given as an N-dimensional row vector.

The optimality condition in Theorem 8.6 enables the development of a gradient descent-type algorithm. The algorithm developed in [33] is summarized in the following.

Algorithm 8.1 Gradient descent-type algorithm for Problem 8.6

```
1: Let $k = 0$, and determine the initial value of $\left\{u_t^{(k)}\right\}_{t=0}^{T-1}$ and the number of the maximal iteration $N_{max}$.
2: Obtain $\left\{p_t^{(k)}\right\}_{t=0}^{T}$ and $\left\{\Lambda_t^{(k)}\right\}_{t=0}^{T}$ using the state and the adjoint equations in (8.57).
3: Update $\left\{u_t^{(k)}\right\}_{t=0}^{T-1}$ as $u_t^{(k+1)} = u_t^{(k)} - \alpha\delta u_t^{(k)}$, where
       $\delta u_{t,m}^{(k)} = \Lambda_{t+1}^{(k)} \frac{\partial P}{\partial u_m}(u_t^{(k)}, t)p_t^{(k)} + \frac{\partial g}{\partial u_m}(p_t^{(k)}, u_t^{(k)}).$   (8.60)
   and $\alpha > 0$ is a small constant.
4: $k \to k + 1$. If $k = N_{\max}$, stop; otherwise, go to step 2.
```

8.5.2 Application to Rebalancing in One-Way Car-Sharing Services

The optimal control problem discussed above is now adapted to the rebalancing problem in one-way car-sharing systems. As mentioned above, the one-way car-sharing services enable users to return vehicles in arbitrary slots, not necessarily the same slots that the users rented. This framework provides convenience to users

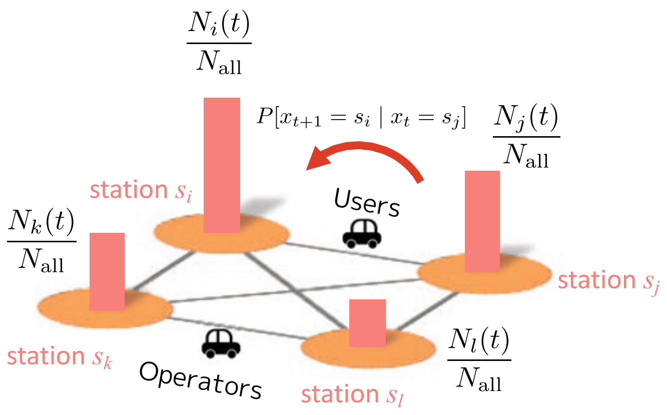

$\frac{N_i(t)}{N_{\text{all}}}$
$P[x_{t+1} = s_i \mid x_t = s_j]$
$\frac{N_j(t)}{N_{\text{all}}}$
station s_i
Users
$\frac{N_k(t)}{N_{\text{all}}}$
station s_j
station s_k
Operators
station s_l
$\frac{N_l(t)}{N_{\text{all}}}$

Fig. 8.13 Setting in One-way Car-sharing system [33]

but requires the rebalancing of vehicles between parking slots for service providers. This issue requires determining an optimal dispatching strategy in the rebalancing of available vehicles. The current framework deals with this issue from the perspective of the probability distribution control problem.

The rebalancing problem is formulated as an optimal control problem as follows (see also Fig. 8.13). We assume that N_{all} vehicles is prepared in the car-sharing service and the number of the vehicles does not change during the operation. Additionally, N parking slots are assumed to be provided, and each parking slot is denoted by $s_1, \ldots, s_N$. Then, $s_1, \ldots, s_N$ forms the state space in the control problem $S = \{s_1, \ldots, s_N\}$. Moreover, we assume that each parking slot is located at the position $z_i \in \mathbb{R}^2$ and $d_{ij} = \|z_i - z_j\|$ denotes the distance between the locations z_i and z_j. Given the vehicles and parking slots, the ith parking slot possesses $N_i(t)$ vehicles at time t. The fractions of vehicles at parking slots determine a probability distribution of a vehicle staying at the ith slots at time t, that is, the probability distribution $p_t \in \mathcal{P}(S)$ possesses the element $(p_t)_i$ given by

$$\mathrm{P}[x_t = s_i] = (p_t)_i := \frac{N_i(t)}{N_{\text{all}}}, \tag{8.61}$$

where x_t denotes the state of a vehicle at time t, and $x_t = s_i$ expresses that the vehicle is locating at or moving to the parking slot s_i. We introduce the desired probability vectors $\{p_{t,d}\}_{t=1}^{T}$ to express the user demand of vehicles in parking slots.

We consider a dynamical model to determine the movements of vehicles between parking slots as a Markov chain (8.50). The Markov chain model of the movements of vehicles is given by

$$p_{t+1} = p_t - L_t p_t - L_t^u p_t, \tag{8.62}$$

where L_t, L_t^u are matrices, and L_t^u is a control input. This equation is obtained by replacing the matrix $P(u_t, t)$ in (8.50) with

$$P(u_t, t) = I - L_t - L_t^u, \tag{8.63}$$

where $u_t = L_t^u$. The matrix L_t is assumed to have the form

$$L_t = \begin{bmatrix} \sum_{i=2}^{N} l_{t,i1} & -l_{t,12} & \dots & -l_{t,1N} \\ -l_{t,21} & \sum_{\substack{i=1 \\ i\neq 2}}^{N} l_{t,i2} & \dots & -l_{t,2N} \\ \vdots & \vdots & \ddots & \vdots \\ -l_{t,N1} & -l_{t,N2} & \dots & \sum_{i=1}^{N-1} l_{t,iN} \end{bmatrix}. \tag{8.64}$$

L_t^u also has the same form as (8.64), where the element $l_{t,ij}$ is replaced with $l_{t,ij}^u$. The element $l_{t,ij}$ in L_t determines the fraction of vehicles moved by users from slot s_j to s_i at time t. Additionally, $l_{t,ij}^u$ of L_t^u models the fraction of vehicles moved in the rebalancing from s_j to s_i. The elements $l_{t,ij}$ of L_t satisfy that $l_{t,ij} \in [0, 1]$ and $\sum_{k=1,k\neq j}^{N} l_{t,kj} \leq 1$ for $j \in \mathbb{Z}_{1:N}$. Given the matrix L_t, the matrix L_t^u has to satisfy conditions for the matrix $I - L_t - L_t^u$ to be a transition matrix. The elements $l_{t,ij}^u$ satisfy that for $i, j \in \mathbb{Z}_{1:N}$, $i \neq j$,

$$l_{t,ij}^u \geq 0, \tag{8.65}$$

and for $j \in \mathbb{Z}_{1:N}$,

$$\sum_{i=1,i\neq j}^{N} l_{t,ij}^u \leq 1 - \sum_{i=1,i\neq j}^{N} l_{t,ij}. \tag{8.66}$$

These conditions on L_t^u guarantee that the matrix $I - L_t - L_t^u$ becomes the transition matrix of the Markov chain.

The cost function of $J(u)$ in the optimal control problem of Problem 8.6 is designed as follows. The stage cost consists of the Wasserstein distance and the control cost $g(p_t, L_t^u)$, where u_t is replaced with L_t^u. The Wasserstein distance is given by $W(p_t, p_{t,d})$ to measure the closeness of the actual and desired distribution of vehicles at parking slots, where the distances between parking slots determine the distances d_{ij} in Definition 8.1. This setting means that the rebalancing strategy is designed to dispatch the vehicles to meet the user demand. Additionally, we introduce the control cost in the rebalancing as follows:

$$g\left(p_t, L_t^u\right) = C_1 \sum_{\substack{i,j=1 \\ j\neq i}}^{N} \left(l_{t,ij}^u p_{t,j}\right)^2 + C_2 g_B\left(L_t^u\right), \tag{8.67}$$

where C_1 and C_2 are non-negative constants, and $g_B : \mathbb{R}^{N\times N} \to \mathbb{R}$ is a barrier function for L_t^u to satisfy the constraints (8.65) and (8.66). In [33], the barrier function $g_B(L_t^u)$ is given by

$$g_B\left(L_t^u\right) = \sum_{\substack{i,j=1\\ i\neq j}}^{N} \frac{1}{l_{t,ij}^u} + \sum_{j=1}^{N} \frac{1}{1 - \sum_{\substack{i=1\\ i\neq j}}^{N} l_{t,ij} - \sum_{\substack{i=1\\ i\neq j}}^{N} l_{t,ij}^u} \tag{8.68}$$

The design of the stage cost with the Wasserstein distance and the control cost including the barrier function yields the rebalancing strategy taking both the user demand and the rebalancing cost into consideration.

We show a numerical example of the optimal control for yielding the rebalancing strategy in the one-way car-sharing services. Consider the setting of the car-sharing service with six parking slots and the total time step $T = 10$. Additionally, the locations of parking slots are randomly determined so that the distances between every pair of the parking slots, d_{ij}, is in [0.5, 5.0]. The desired probability distribution $p_{t,d}$ and the matrices L_t ($t \in \mathbb{Z}_{1:T}$) are also randomly generated. Moreover, the following parameters are used; $C_t = 100, 0.1$, $C_2 = 10^{-6}$, $\beta = 1$. The parameters $\alpha = 0.0002$ and $N_{\max} = 20,0000$ are used in Algorithm 8.1. In the following, we refer to the cases of $C_1 = 100$ and $C_1 = 0.1$ as cases 1 and 2, respectively. Case 1 focuses on reducing the control cost, i.e., the rebalancing strategies prioritize reducing the cost for the dispatching in the rebalancing rather than the matching between the actual numbers of vehicles and the desired numbers determined by user demand. Conversely, case 2 prioritizes meeting the actual and required distributions of vehicles by finding the optimal strategy. Figure 8.14 shows the values of the Wasserstein distances between p_t and $p_{t,d}$ on the horizon $t \in \mathbb{Z}_{1:T}$ for cases 1 and 2. Case 2 has a stronger focus on reducing Wasserstein costs than case 1. Therefore, the values of the Wasserstein distances in case 2 are smaller than in case 1. In case 1, we obtained $\sum_{t=1}^{T} W\left(p_t, p_{t,d}\right) = 3.69$ and $\sum_{t=0}^{T-1} g\left(p_t, L_t^u\right) = 0.006$. Case 2 yields $\sum_{t=1}^{T} W\left(p_t, p_{t,d}\right) = 3.08$ and $\sum_{t=0}^{T-1} g\left(p_t, L_t^u\right) = 0.026$. These results imply that the value of C_1 can adjust preferences in yielding the rebalancing strategy from the optimal control and that the Wasserstein distance between the actual distribution of

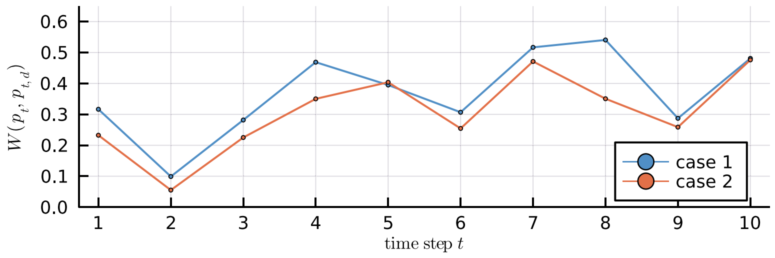


Fig. 8.14 Values of Wasserstein distances $W(p_t, p_{t,d})$ for $t \in \mathbb{Z}_{1:T}$ [33]

vehicles and the desired distribution determined by user demand can be reduced by choosing a sufficiently small value of C_1.

8.6 Optimization of One-Way Car-Sharing Services via DC Program

One-way car-sharing services can be considered as a special class of buffer networks. Recently, the study of buffer networks has emerged in which the nodes among them behave as buffers to exchange the inflow/outflow with their neighbors. Several applications, such as environment [54], power [71], and network-on-chip design [63], have triggered the interest for conducting this type of study. Moreover, there are emerging applications in the mobility systems domain, such as the optimization of traffic flows [13, 20, 73] and the design of efficient and effective mobility-on-demand systems [5, 41]. A standard model used to study buffer networks is the positive linear system model [64]. Using this model, a number of fundamental control and optimization problems have been studied. An example is the stability analysis studied in [14, 52]. Moreover, the positive linear system model enables the study of control and optimization problems [59]. Further, the robust control problem has been studied in [65, 68]. The goal of this section is to show that the DC (Difference of Convex Functions) program is an adequate mathematical tool to address the problem of efficient control.

8.6.1 Problem Formulation

8.6.1.1 Service Model

Here, we consider a one-way car-sharing service in which the stations providing parking slots enable customers to rent vehicles at any stations. Let n be the number of stations in an area. We assume that the structure of the service network is described by the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, where $\mathcal{V} = \{v_1, \ldots, v_n\}$ represents the set of n stations within the service and $\mathcal{E} = \{e_1, \ldots, e_m\} \subseteq \mathcal{V} \times \mathcal{V}$ represents the set of directed edges indicating possible usage between stations. We suppose that there exist two special sets of stations that serve as origins (i.e., the stations having an empty in-neighborhood node) and destinations (i.e., the stations having an empty out-neighborhood node). We let $\mathcal{V}_o = \{1, \ldots, |\mathcal{V}_o|\}$ and $\mathcal{V}_d$ denote the set of origins and destinations of the car-sharing service, respectively.

Inspired by [40], we construct a mathematical model of the system of the one-way car-sharing service as follows. Let $x_i(t) \in \mathbb{R}_+$ $(i \in \mathcal{N})$ denote the expectation of the number of vehicles parking at station i and at time $t \geq 0$. Moreover, let $u_{ij}(t) \in \mathbb{R}_+$ $((i, j) \in \mathcal{E})$ be the expectation of the number of customers who travel from station i to j at time t. Let $f_i^{\text{in}} \in \mathbb{R}_+$ $(f_i^{\text{out}} \in \mathbb{R}_+)$ be the expectation of the number of vehicles moving to (from) this area from (to) other areas. Then, we suppose that x_i obeys the following differential equation

$$\frac{dx_i}{dt} = \begin{cases} f_i^{\text{in}} - \sum_{j \in \mathcal{N}_i^{\text{out}}} u_{ij}, & \text{if } \ i \in \mathcal{V}_o, \\ \sum_{j \in \mathcal{N}_i^{\text{in}}} u_{ji} - \sum_{j \in \mathcal{N}_i^{\text{out}}} u_{ij}, & \text{if } \ i \notin \mathcal{V}_o \cup \mathcal{V}_d, \\ \sum_{j \in \mathcal{N}_i^{\text{in}}} u_{ji} - f_i^{\text{out}}, & \text{if } \ i \in \mathcal{V}_d, \end{cases} \tag{8.69}$$

where $\mathcal{N}_i^{\text{in}} = \{j \in \mathcal{V} : (j, i) \in \mathcal{E}\}$ denotes the set of in-neighborhoods of station i and $\mathcal{N}_i^{\text{out}} = \{j \in \mathcal{V} : (i, j) \in \mathcal{E}\}$ denotes the set of out-neighborhoods of station i. Further, we assume

$$f_i^{\text{out}} = \beta_i x_i, \tag{8.70}$$

where $\beta = \{\beta_i\}_{i \in \mathcal{V}_d}$ are the parameters to be tuned by the designer of the service.

We adopt the following model of the demand of customers that can change with prices. Assume that the expectation of the demand is $\bar{u}_{ij}(t) \in \mathbb{R}_+$ when the price is $\bar{p}_{ij}(t) \in \mathbb{R}_+$ and that the change of the demand is governed with an affine model around this point. Let $p_{ij} \in \mathbb{R}_+$ be the price for traveling from stations i to j, and let $\delta_{ij} \in \mathbb{R}_+$ be the price elasticity. Then, we consider the affine price model

$$u_{ij} = \bar{u}_{ij} - \delta_{ij}(p_{ij} - \bar{p}_{ij}). \tag{8.71}$$

As a pricing strategy, we suppose that the price p_{ij} is adjusted according to the number x_i of vehicles at station i as

$$p_{ij} = \hat{p}_{ij} - w_{ij} x_i, \tag{8.72}$$

where $\hat{p}_{ij} \in \mathbb{R}_+$ and $w_{ij} \in \mathbb{R}_+$ are design parameters. Here, we set $\hat{p}_{ij} = \bar{p}_{ij} + \bar{u}_{ij}/\delta_{ij}$. Then, from (8.71) and (8.72), the demand model is reduced to

$$u_{ij} = \delta_{ij} w_{ij} x_i. \tag{8.73}$$

We suppose that $\delta = \{\delta_{ij}\}_{(i,j) \in \mathcal{E}}$ can also be tuned by the operator of the service.

To formulate our optimization problem for designing the car-sharing service, we rewrite the model described above into a standard linear system. Let

$$[A_{\mathcal{G}}]_{ij} = \begin{cases} w_{ji}, & \text{if } (j, i) \in \mathcal{E}, \\ 0, & \text{otherwise.} \end{cases}$$

be the adjacency matrix of the graph $\mathcal{G}$. To measure the performance of the buffer network, we adopt the output $y = [x^\top \ \alpha u^\top]^\top$, where $\alpha > 0$ is a weight constant and $u \in \mathbb{R}_+^{n\times n}$ includes the information of the edges. If we define

$$B_{ij} = \begin{cases} \beta_i, & \text{if } i = j, \\ 0, & \text{otherwise,} \end{cases} \quad D_{ij} = \begin{cases} \delta_{ji}, & \text{if } (i, j) \in \mathcal{E}, \\ 0, & \text{otherwise,} \end{cases}$$

then the dynamic model can be expressed as

$$\Sigma : \begin{cases} \dfrac{dx}{dt} = \Big(D \odot A_{\mathcal{G}} - \operatorname{diag}\big(1_n^\top (D \odot A_{\mathcal{G}})\big) - B\Big)x + G^{\text{in}} f^{\text{in}}, \\ y = G^{\text{out}}(\delta)x, \end{cases}$$

where the vector f^{in} and the matrices G^{in}, $G^{\text{out}}(\delta)$ are defined by $f^{\text{in}}{=}[f_1^{\text{in}} \cdots f_{|\mathcal{V}_o|}^{\text{in}}]^\top$ and

$$G^{\text{in}} = \begin{bmatrix} I_{|\mathcal{V}_o|} \\ O_{n-|\mathcal{V}_o|,|\mathcal{V}_o|} \end{bmatrix}, \ G^{\text{out}}(\delta) = \begin{bmatrix} I_n \\ \alpha H(\delta) \end{bmatrix}, \tag{8.74}$$

while the matrix $H(\delta)$ is defined by

$$H(\delta)_{\ell i} = \begin{cases} \delta_{e_\ell} w_{e_\ell}, & \text{if } i = e_\ell(1), \\ 0, & \text{otherwise.} \end{cases}$$

We remark that, as G^{in} and $G^{\text{out}}(\delta)$ are non-negative matrices and $D \odot A_{\mathcal{G}} - \operatorname{diag}\big(1_n^\top (D \odot A_{\mathcal{G}})\big) - B$ is the Metzler matrix, following the definition in [18], the dynamic model (8.74) is a positive linear system [59].

8.6.1.2 Optimization Problem

As stated in the previous section, the parameters β_i and δ_{ij} are assumed to be tunable by the operator to improve the performance of the car-sharing system. We consider the situation in which adopting a specific parameter β and δ requires a cost $L(\beta, \delta) = \sum_{i\in\mathcal{V}_d} g_i(\beta_i) + \sum_{(i,j)\in\mathcal{E}} h_{ij}(\delta_{ij})$. We allow the following interval constraint:

$$0 < \beta_i \le \bar{\beta}_i, \ 0 < \delta_{ij} \le \bar{\delta}_{ij}. \tag{8.75}$$

In this section, we adopt the H^∞ norms to measure the performance of the buffer network. If system Σ is stable, then the H^∞ norm of the system is defined as $\|\Sigma\|_\infty = \sup_{w\in\mathcal{L}^2} \|\Omega * w\|_2/\|w\|_2$, where $\Omega \neq 0$ and $*$ denotes a convolution product and $\mathcal{L}^2 = \{f : [0, \infty) \to \mathbb{R}^n \mid \int_0^\infty \|f(t)\|^2 dt < \infty\}$ denotes the space of Lebesgue measurement functions.

We are now ready to state the optimization problem studied in this section.

Problem 8.7 (H^∞ *norm-constrained optimization*) Given the desired H^∞ performance $\gamma_\infty > 0$. Find the parameters β and δ minimizing the parameter tuning cost $L(\beta, \delta)$, under the constraint that the parameter constraints (8.75) are satisfied, Σ is stable, and the system requirement $\|\Sigma\|_\infty < \gamma_\infty$ is satisfied.

8.6.2 DC Programming

We present the solutions to the H^2 and H^∞ norm-constrained optimization problems in terms of DC (Difference of Convex Functions) programming [31, 49]. To state the main results, we begin this section by introducing the preliminary knowledge of posynomials, DC functions, and DC program.

Definition 8.2 Let $v_1, \ldots,$ and v_n denote n real positive variables. We state that a real function $g(v)$ is a *monomial* if $c > 0$ and $a_1, \ldots, a_n \in \mathbb{R}$ such that $g(v) = cv_1^{a_1} \cdots v_n^{a_n}$. We state that a real function $f(v)$ is a *posynomial* [4] if f is the sum of the monomials of v.

The following lemma shows the log-convexity of posynomials [4].

Lemma 8.1 *If* $f: \mathbb{R}^n_{++} \to \mathbb{R}_{++}$ *is a posynomial function, then the log-transformed function* $F: \mathbb{R}^n \to \mathbb{R}: w \mapsto \log[f(\exp[w])]$ *is convex.*

The property shown in Lemma 8.1 enables us to establish a relationship with a general class of mathematical programming that deals with the difference between two convex functions, called DC programming.

Definition 8.3 (*DC functions* [31]) Let C be a convex subset of $\mathbb{R}^n$. A real-valued function $f : C \to \mathbb{R}$ is called a *DC function* on C if there exist two convex functions $g, h : C \to \mathbb{R}$ such that f can be expressed in the form $f(x) = g(x) - h(x)$.

Any continuous function can be approximated by a DC function with the desired accuracy. In fact, based on decomposition methods [31], it is possible to transform a nonconvex optimization problem into a DC programming problem.

Definition 8.4 (*DC programming problem* [31]) Programming problems dealing with DC functions are called *DC programming problems*. Let C be a closed convex subset of $\mathbb{R}^n$, and the general form of DC programming problem considered in this brief is

$$\begin{aligned} &\underset{x \in C}{\text{minimize}}\ f_0(x) \\ &\text{subject to } f_i(x) \le 0,\ i = 1, \ldots, m, \end{aligned}$$

where $f_0(x) = g_0(x) - h_0(x)$ and $f_i(x) = g_i(x) - h_i(x)$ are the differences of the two convex functions.

To optimally solve the DC programming problem, we can choose the branch-and-bound type and outer-approximation algorithms [15], which lead to more efficient procedures.

We place the following assumption on the cost functions.

Assumption 8.6 The functions $g_i(\beta_i)$ and $h_{ij}(\delta_{ij})$ are posynomials for all i and j.

The following theorem states that Problem 8.7 reduces to a DC program. For the proof, readers are referred to [78].

Theorem 8.7 ([78]) *Under the aforementioned assumptions and lemma, the solution of Problem 8.7 is given by*

$$B = \exp[\gamma],\ \ D = \exp[\eta], \tag{8.76}$$

where γ and η are the solution of the DC programming

$$\begin{aligned}
\underset{\substack{\gamma,\eta\in\mathbb{R} \\ \mu,\nu,\xi,\zeta\in\mathbb{R}^n_{++}}}{\text{minimize}}\quad & \log[L(\exp[\gamma],\exp[\eta])] \\
\text{subject to}\quad & \log[G^{\text{out}}(\exp[\eta])\xi] - \log[\gamma_\infty \nu] < 0, \\
& \log[(\exp[\eta]\odot A_{\mathcal{G}})\xi + G^{\text{in}}\mu] - \\
& \log[(\text{diag}(1_n^\top \exp[\eta]\odot A_{\mathcal{G}}) + \exp[\gamma])\xi] < 0, \\
& \log[G^{\text{in}\top}\zeta] - \log[\gamma_\infty \mu] < 0, \\
& \log[(\exp[\eta]\odot A_{\mathcal{G}})^\top \zeta + G^{\text{out}}(\exp[\eta])^\top \nu] - \\
& \log[(\text{diag}(1_n^\top \exp[\eta]\odot A_{\mathcal{G}}) + \exp[\gamma])^\top \zeta] < 0, \\
& \log[\exp[\gamma]] - \log[\exp[\bar{\gamma}]] < 0, \\
& \log[\exp[\eta]] - \log[\exp[\bar{\eta}]] < 0.
\end{aligned}$$

8.6.3 Numerical Simulation

In this simulation, we construct a network with 30 nodes (see Fig. 8.15). We configure the network structure to contain one origin, one destination, and 60 edges. For simplicity, we set $\alpha = 0$. We also assume that $g_i(\beta_i) = \beta_i$ and $h_{ij}(\delta_{ij}) = \delta_{ij}$, which indicates that the increment of the variable has a positive correlation with the tuning cost.

We solve the H^∞ norm-constrained optimization problem for various cost bounds $\bar{L}$. To solve the DC program obtained from the theorem, we employ the proximal bundle method presented in [15]. As a baseline approach, we take the optimization methodology based on geometric programming [59]. Specifically, we reduce the DC problem to a geometric program by imposing the constraint $\delta_{ij} = \delta_{ik}$ for all k, which enables solving the optimization problem with geometric program-

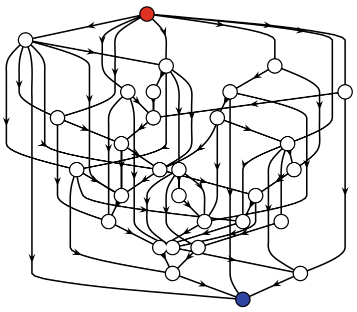


Fig. 8.15 Buffer network comprising 30 nodes (red: origin, blue: destination)

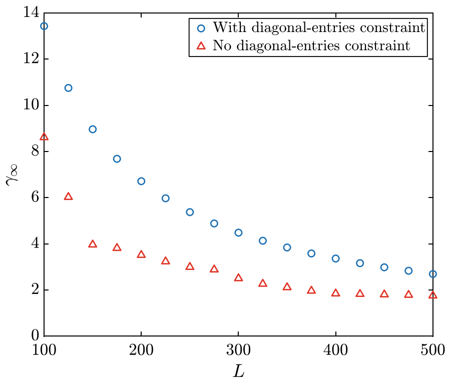


Fig. 8.16 Optimized H^∞ norm versus the cost constraints

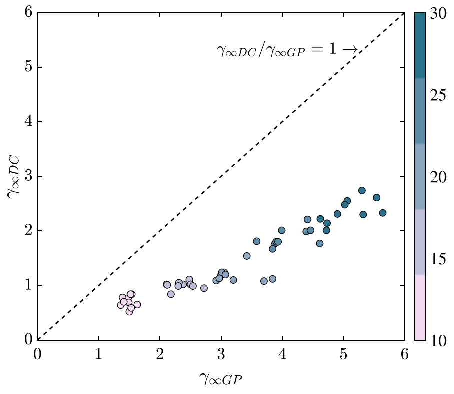


Fig. 8.17 Plot of the $\gamma_{\infty DC}/\gamma_{\infty GP}$ ratio versus the size of network. $\gamma_{\infty DC}$: the optimized H^∞ norm solved by Theorem 8.7. $\gamma_{\infty GP}$: optimal H^∞ norm solved by geometric programming (colorbar: size of network)

ming. In Fig. 8.16, the triangles show the optimized γ_∞ for various $\bar{L}$, and the circles show the optimal solution from geometric programming. We note that our DC programming-based results showed higher performance (Fig. 8.17).

8.7 Conclusion

This chapter discussed the problem of the uneven distribution in one-way car-sharing services. The authors discussed this issue from system control approaches as follows.

In Sect. 8.2, we designed a common decentralized policy for dynamic prices to solve the uneven distribution of vehicles in car-sharing services. We focused on the demand shift of users to less expensive links and modeled it as a stochastic process. Then, an optimization problem was solved to determine the best parameters of the dynamic pricing policy. The effectiveness of the model was verified using the traffic simulator SOUND (Simulation On Urban road Network with Dynamic route choice).

Section 8.3 introduced the sparse optimal control theory for networked control systems and then examined the engineering significance of the problem setting through the rebalancing problem in a one-way car-sharing service. We hope this chapter sparks the reader's interest in the sparse optimal control of network systems.

Section 8.4 reviewed distributed rebalancing control for cooperative car-sharing service. The service providers operate a sharing service in which the number of dead-head vehicles is as small as possible by dispatching vehicles to match user demand. To solve the control problem in a distributed manner, we presented a consensus-based primal-dual method with event-triggered communication. We showed that the estimation of each provider converges to an optimal solution of the rebalancing control problem.

Section 8.5 discussed an optimal control of probability distributions of Markov decision processes and its application to the rebalancing operations of one-way car-sharing services. We formulated the optimal control problem of probability distributions as the minimization of the Wasserstein distance between the controlled and desired distributions. This formulation was applied to the rebalancing problem, which requires the minimization of the difference between the distributions of the availability of vehicles and the user demand for required vehicles. We showed that the control formulation provides rebalancing strategies that match availability and user demand while taking rebalancing costs into consideration.

In Sect. 8.6, we studied the H^∞ norm-constrained optimization problem for one-way car-sharing services. Using the log-log convexity property of posynomials, we showed that the optimization problem reduces to a standard DC program. We also presented numerical simulations to illustrate the numerical scalability and effectiveness of the proposed optimization framework. There are several future research directions to be explored. One of them is the consideration of the L^1 induced norm. Another direction is to apply our results to other socio-technical systems.

These results show that the control system approach is promising for mobility innovation.

References

1. T. Adachi, N. Hayashi, S. Takai, Distributed gradient descent method with edge-based event-driven communication for non-convex optimization. IET Control Theory Appl. **15**(12), 1588–1598 (2021)
2. I.M. Balci, E. Bakolas, Covariance steering of discrete-time stochastic linear systems based on Wasserstein distance terminal cost. IEEE Control Syst. Lett. **5**(6), 2000–2005 (2020)
3. M. Barth, M. Todd, Simulation model performance analysis of a multiple station shared vehicle system. Transp. Res. Part C: Emerg. Technol. **7**(4), 237–258 (1999)
4. S. Boyd, S.-J. Kim, L. Vandenberghe, A. Hassibi, A tutorial on geometric programming. Optim. Eng. **8**(1), 67–127 (2007)
5. G.C. Calafiore, C. Bongiorno, A. Rizzo, A control-oriented model for mobility on demand systems, in *Proceedings of the 57th IEEE Conference on Decision and Control* (2018), pp. 999–1004
6. G.C. Calafiore, C. Bongiorno, A. Rizzo, A robust MPC approach for the rebalancing of mobility on demand systems. Control Eng. Pract. **90**, 169–181 (2019)
7. X. Cao, T. Başar, Decentralized online convex optimization with event-triggered communications. IEEE Trans. Signal Process. **69**, 284–299 (2021)
8. G. Carnevale, I. Notarnicola, L. Marconi, G. Notarstefano, Triggered gradient tracking for asynchronous distributed optimization. Automatica **147**, 110726 (2023)
9. C.B. Casady, Customer-led mobility: a research agenda for Mobility-as-a-Service (MaaS) enablement. Case Stud. Transp. Policy **8**(4), 1451–1457 (2020)
10. T.H. Chang, M. Hong, X. Wang, Multi-agent distributed optimization via inexact consensus ADMM. IEEE Trans. Signal Process. **63**(2), 482–497 (2015)
11. T.H. Chang, A. Nedić, A. Scaglione, Distributed constrained optimization by consensus-based primal-dual perturbation method. IEEE Trans. Autom. Control **59**(6), 1524–1538 (2014)
12. Y. Chen, T.T. Georgiou, M. Pavon, Optimal transport over a linear dynamical system. IEEE Trans. Autom. Control **62**(5), 2137–2152 (2017)
13. G. Como, K. Savla, D. Acemoglu, M.A. Dahleh, E. Frazzoli, Robust distributed routing in dynamical networks-part I: locally responsive policies and weak resilience. IEEE Trans. Autom. Control **58**(2), 317–332 (2013)
14. S. Coogan, M. Arcak, A compartmental model for traffic networks and its dynamical behavior. IEEE Trans. Autom. Control **60**(10), 2698–2703 (2015)
15. W. de Oliveira, Proximal bundle methods for nonsmooth DC programming. J. Global Optim. **75**(2), 523–563 (2019)
16. S. Diamond, S. Boyd, CVXPY: a Python-embedded modeling language for convex optimization. J. Mach. Learn. Res. **17**(83), 1–5 (2016)
17. A. Falsone, K. Margellos, S. Garatti, M. Prandini, Dual decomposition for multi-agent distributed optimization with coupling constraints. Automatica **84**, 149–158 (2017)
18. L. Farina, S. Rinaldi, *Positive Linear Systems: Theory and Applications* (Wiley-Interscience, 2000)
19. F. Ferrero, G. Perboli, M. Rosano, A. Vesco, Car-sharing services: an annotated review. Sustain. Cities Soc. **37**, 501–518 (2018)
20. P. Grandinetti, C. Canudas-de Wit, F. Garin, Distributed optimal traffic lights design for large-scale urban networks. IEEE Trans. Control Syst. Technol. **27**(3), 950–963 (2019)
21. Z. Haider, A. Nikolaev, J.E. Kang, C. Kwon, Inventory rebalancing through pricing in public bike-sharing systems. Eur. J. Oper. Res. **270**, 103–117 (2018)
22. A. Halder, E.D. Wendel, Finite horizon linear quadratic Gaussian density regulator with Wasserstein terminal cost, in *Proceedings of the 2016 American Control Conference* (2016), pp. 7249–7254
23. K. Hayashi, M. Nagahara, T. Tanaka, A user's guide to compressed sensing for communications systems. IEICE Trans. Commun. **E96.B**(3), 685–712 (2013)
24. N. Hayashi, K. Sakurama, Communication-aware distributed rebalancing for cooperative car-sharing service. IET Control Theory Appl. **17**(7), 850–867 (2022)

25. N. Hayashi, T. Sugiura, Y. Kajiyama, S. Takai, Event-triggered consensus-based optimization algorithm for smooth and strongly convex cost functions, in *Proceedings of the 57th IEEE Conference on Decision and Control* (2018), pp. 2120–2125
26. N. Hayashi, T. Sugiura, Y. Kajiyama, S. Takai, Distributed event-triggered algorithm for unconstrained convex optimization over weight-balanced directed networks. IET Control Theory Appl. **14**(2), 253–261 (2020)
27. N. Hayashi, T. Ushio, Application of a consensus problem to fair multi-resource allocation in real-time system, in *Proceedings of the 47th IEEE Conference on Decision and Control* (2008), pp. 2450–2455
28. N. Hayashi, T. Ushio, F. Harada, A. Ohno, Consensus problem of multi-agent systems with non-linear performance functions. IEICE Trans. Fundam. **90**(10), 2261–2264 (2007)
29. W.P.M.H. Heemels, K.H. Johansson, P. Tabuada, An introduction to event-triggered and self-triggered control, in *Proceedings of the 51st IEEE Annual Conference on Decision and Control* (2012), pp. 3270–3285
30. D.A. Hensher, C. Mulley, C. Ho, Y. Wong, G. Smith, J.D. Nelson, *Understanding Mobility as a Service (MaaS): Past, Present and Future* (Elsevier, 2020)
31. R. Horst, N.V. Thoai, DC programming: overview. J. Optim. Theory Appl. **103**(1), 1–43 (1999)
32. K. Hoshino, Finite-horizon control of nonlinear discrete-time systems with terminal cost of Wasserstein distance, in *Proceedings of the 59th IEEE Conference on Decision and Control* (2020), pp. 4268–4274
33. K. Hoshino, K. Sakurama, Probability distribution control of finite-state Markov chains with Wasserstein costs and application to operation of car-sharing services, in *Proceedings of the 60th IEEE Conference on Decision and Control* (2021), pp. 6634–6639
34. i-Transport Lab. Co., Ltd. SOUND Ver.5, a traffic simulator for wide area road networks. https://www.i-transportlab.jp/en/index/products/sound/
35. T. Ikeda, K. Kashima, Sparsity-constrained controllability maximization with application to time-varying control node selection. IEEE Control Syst. Lett. **2**(3), 321–326 (2018)
36. T. Ikeda, K. Kashima, On sparse optimal control for general linear systems. IEEE Trans. Autom. Control **64**(5), 2077–2083 (2019)
37. T. Ikeda, K. Kashima, Optimal time-varying topology for network systems, in *Proceedings of the 13th Asian Control Conference* (2022), pp. 1926–1931
38. T. Ikeda, K. Kashima, Sparse control node scheduling in networked systems based on approximate controllability metrics. IEEE Trans. Control Netw. Syst. **9**(3), 1166–1177 (2022)
39. T. Ikeda, M. Nagahara, K. Kashima, Maximum hands-off distributed control for consensus of multiagent systems with sampled-data state observation. IEEE Trans. Control Netw. Syst. **6**(2), 852–862 (2019)
40. T. Ikeda, K. Sakurama, K. Kashima, Multiple sparsity constrained control node scheduling with application to rebalancing of mobility networks. IEEE Trans. Autom. Control **67**(8), 4314–4321 (2022)
41. S. Illgen, M. Höck, Literature review of the vehicle relocation problem in one-way car sharing networks. Transp. Res. Part B: Methodol. **120**, 193–204 (2019)
42. K. Ito, T. Ikeda, K. Kashima, Sparse optimal stochastic control. Automatica **125**, 109438 (2021)
43. A. Jadbabaie, J. Lin, A.S. Morse, Coordination of groups of mobile autonomous agents using nearest neighbor rules. IEEE Trans. Autom. Control **48**(6), 988–1001 (2003)
44. D. Jorge, G. Correia, Carsharing systems demand estimation and defined operations: a literature review. Eur. J. Transp. Infrastruct. Res. **13**(3), 201–220 (2013)
45. D. Jorge, G. Molnar, G.H. de Almeida Correia, Trip pricing of one-way station based carsharing networks with zone and time of day price variations. Transp. Res. Part B: Methodol. **81**, 461–482 (2015)
46. Y. Kajiyama, N. Hayashi, S. Takai, Distributed subgradient method with edge-based event-triggered communication. IEEE Trans. Autom. Control **63**(7), 2248–2255 (2018)
47. T. Kamatani, Y. Nakata, S. Arai, Dynamic pricing method to maximize utilization of one-way car-sharing service, in *Proceeding of the 2019 IEEE International Conference on Agents* (2019), pp. 65–68

48. A.G. Kek, R.L. Cheu, Q. Meng, C.H. Fung, A decision support system for vehicle relocation operations in carsharing systems. Transp. Res. Part E: Logist. Transp. Rev. **45**(1), 149–158 (2009)
49. H.A. Le Thi, T. Pham Dinh, DC programming and DCA: thirty years of developments. Math. Program. **169**(1), 5–68 (2018)
50. C. Liu, H. Li, Y. Shi, D. Xu, Distributed event-triggered gradient method for constrained convex minimization. IEEE Trans. Autom. Control **65**(2), 778–785 (2020)
51. F. Lorig, J.A. Persson, A. Michielsen, Simulating the impact of shared mobility on demand: a study of future transportation systems in Gothenburg. Sweden. Int. J. Intell. Transp. Syst. Res. **21**(1), 129–144 (2023)
52. E. Lovisari, G. Como, K. Savla, Stability of monotone dynamical flow networks, in *Proceedings of the 53rd IEEE Conference on Decision and Control* (2014), pp. 2384–2389
53. M. Mesbahi, M. Egerstedt, *Graph Theoretic Methods in Multiagent Networks* (Princeton University Press, 2010)
54. A. Michez, H. Piégay, P. Lejeune, H. Claessens, Multi-temporal monitoring of a regional riparian buffer network ($>$12,000km) with LiDAR and photogrammetric point clouds. J. Environ. Manage. **202**, 424–436 (2017)
55. S. Mizokami, T. Maruyama, J. Hashimoto, T. Mori, D. Sunaga, Research and development on the possibility of introducing a one-way car-sharing system and new ways to utilize road space, in *New Road Technology Conference* (2019). In Japanese
56. M. Nagahara, D.E. Quevedo, D. Nesic, Maximum hands-off control: a paradigm of control effort minimization. IEEE Trans. Autom. Control **61**(3), 735–747 (2016)
57. A. Nedić, A. Olshevsky, M.G. Rabbat, Network topology and communication-computation tradeoffs in decentralized optimization. Proc. IEEE **106**(5), 953–976 (2018)
58. A. Nedić, A. Ozdaglar, Distributed subgradient methods for multi-agent optimization. IEEE Trans. Autom. Control **54**(1), 48–61 (2009)
59. M. Ogura, M. Kishida, J. Lam, Geometric programming for optimal positive linear systems. IEEE Trans. Autom. Control **65**(11), 4648–4663 (2020)
60. T. Ohtsuka, T. Ikeda, K. Kashima, Matrix pontryagin principle approach to controllability metrics maximization under sparsity constraints (2022). arXiv:2203.12828
61. M. Parkin, *Economics* (Pearson, 2016)
62. G. Peyré, M. Cuturi, Computational optimal transport: with applications to data science. Found. Trends Mach. Learn. **11**(5–6), 355–607 (2019)
63. C. Pu, W. Cui, J. Wu, J. Yang, Bufferless transmission in complex networks. IEEE Trans. Circuits Syst. II Express Briefs **65**(7), 893–897 (2018)
64. A. Rantzer, M.E. Valcher, A tutorial on positive systems and large scale control, in *Proceedings of the 57th IEEE Conference on Decision and Control* (2019), pp. 3686–3697
65. M. Rotkowitz, R. Cogill, S. Lall, Convexity of optimal control over networks with delays and arbitrary topology. Int. J. Syst. Control Commun. **2**(1–3), 30–54 (2010)
66. K. Sakurama, Optimal control and station relocation of vehicle-sharing systems with distributed dynamic pricing. IEEE Open J. Intell. Transp. Syst. **4**, 393–405 (2023)
67. K. Sakurama, T. Aoki, Distributed dynamic pricing for car-sharing systems with stochastic demand shift, in *Proceedings of the 25th IEEE International Intelligent Transportation Systems Conference* (2022), pp. 1970–1975
68. K. Savla, G. Como, M.A. Dahleh, Robust network routing under cascading failures, in *Proceedings of the 53rd IEEE Conference on Decision and Control* (2014), pp. 2889–2894
69. S.L. Smith, M. Pavone, M. Schwager, E. Frazzoli, D. Rus, Rebalancing the rebalancers: optimally routing vehicles and drivers in mobility on-demand systems, in *Proceedings of the American Control Conference* (2013), pp. 2362–2367
70. K. Sumida, K. Sakurama, T. Aoki, Demand shift model of a one-way car-sharing system with real-time pricing, in *Proceedings of the 24th IEEE International Intelligent Transportation Systems Conference* (2021), pp. 3271–3277
71. N. Vafamand, M.H. Khooban, T. Dragicevic, F. Blaabjerg, Networked fuzzy predictive control of power buffers for dynamic stabilization of DC microgrids. IEEE Trans. Ind. Electron. **66**(2), 1356–1362 (2019)

72. L. Wang, W. Ma, Pricing approach to balance demands for one-way car-sharing systems, in *Proceedings of the 22nd IEEE Intelligent Transportation Systems Conference* (2019), pp. 1697–1702
73. X. Wu, Y. Zhou, The optimal reverse channel choice under supply chain competition. Eur. J. Oper. Res. **259**(1), 63–66 (2017)
74. R. Xie, W. Wei, Q. Wu, T. Ding, S. Mei, Optimal service pricing and charging scheduling of an electric vehicle sharing system. IEEE Trans. Veh. Technol. **69**(1), 78–89 (2020)
75. M. Xu, Q. Meng, Z. Liu, Electric vehicle fleet size and trip pricing for one-way carsharing services considering vehicle relocation and personnel assignment. Transp. Res. Part B: Methodol. **111**, 60–82 (2018)
76. T. Yang, X. Yi, J. Wu, Y. Yuan, D. Wu, Z. Meng, Y. Hong, H. Wang, Z. Lin, K.H. Johansson, A survey of distributed optimization. Annu. Rev. Control **47**, 278–305 (2019)
77. R. Zhang, F. Rossi, M. Pavone, Analysis, control, and evaluation of mobility-on-demand systems: a queueing-theoretical approach. IEEE Trans. Control Netw. Syst. **6**(1), 115–126 (2019)
78. C. Zhao, K. Sakurama, M. Ogura, Optimization of buffer networks via DC programming. IEEE Trans. Circuits Syst. II Express Briefs **70**(2), 606–610 (2023)
79. M. Zhu, S. Martínez, On distributed convex optimization under inequality and equality constraints. IEEE Trans. Autom. Control **57**(1), 151–164 (2012)

Chapter 9
Algorithms for Future Mobility Society

Yasushi Kawase, Kazuhisa Makino, and Hanna Sumita

9.1 Introduction

Algorithms for mobility have been studied for decades in the areas such as operations research, optimization, and graph theory. Typical examples include algorithms for the shortest path, the minimum spanning tree, the maximum flow, the traveling salesman, the Steiner tree, and the facility location problems. These algorithms are used to solve real-world problems in networks such as traffic, information, communication, and social networks. For example, automotive navigation systems are based on shortest path algorithms [48]. Mobility algorithms are fundamental to our daily lives.

In the fields of optimization and computer science, a lot of effort has been made to construct efficient algorithms for various types of optimization problems. However, it is often difficult to model real-world problems as traditional optimization ones, and there exist computational barriers for problems such as NP-hardness, which is briefly described in Sect. 9.2.

As an example of the difficulty of modeling, we may not know the complete information needed to solve the optimization problem in advance. Suppose someone is driving to a restaurant and wants to arrive as early as possible. In this scenario, the next direction in the route map is sequentially selected considering the dynamically changing traffic conditions. Sometimes, we must choose an action in each step based on the current information without knowing the complete information, which will

Y. Kawase
University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan
e-mail: kawase@mist.i.u-tokyo.ac.jp

K. Makino
Kyoto University, Kitashirakawa Oiwake-cho, Sakyo-ku, Kyoto 606-8502, Japan
e-mail: makino@kurims.kyoto-u.ac.jp

H. Sumita (✉)
Tokyo Institute of Technology, 2-12-1 Ookayama, Meguro-ku, Tokyo 152-8550, Japan
e-mail: sumita@c.titech.ac.jp

K. Ikeda et al. (eds.), *Advanced Mathematical Science for Mobility Society*,
https://doi.org/10.1007/978-981-99-9772-5_9

be revealed piece by piece in the future. Such sequential problems are called *online problems*, whereas traditional problems, in which the entire input is given in advance, are called *offline problems*. As another example, real-world problems often have multiple agents, each of whom aims to optimize their own objectives. However, in traditional optimization problems, only one agent makes decisions. In the former case, achieving an outcome that is satisfactory for everyone is desirable. Problems with multiple agents have been studied in economics and game theory under the name of *mechanism design*, which aims to design economic mechanisms for the desired objectives, where agents behave rationally and self-interested.

Algorithms for mobility have been studied in a joint project between Kyoto University and Toyota Motor Corporation, titled "Advanced Mathematical Science for Mobility Society," which started in 2020, with a focus on traditional optimization, online optimization, and mechanism design. In this chapter, we briefly introduce online optimization and mechanism design and present some of the results obtained from this project.

The remainder of this chapter is organized as follows. In Sect. 9.2, we describe approaches to solve traditional combinatorial optimization problems by taking the shortest path and traveling salesman problems as examples. Subsequently, we discuss our results for the reallocation problem. In Sect. 9.3, we introduce the framework for online optimization problems and present a competitive analysis. In particular, we focus on the problems called the online traveling salesman and the online dial-a-ride problems, which occur naturally in a mobility society. Subsequently, we describe our results for the online machine scheduling problem, which is a generalization of the two aforementioned online optimization problems. Finally, in Sect. 9.4, we describe the fair division of resources, which is one of the main topics that appear in multi-agent systems. We introduce solution concepts for fair division, such as envy-freeness and proportionality, and present known results for the envy-free and proportional division of divisible resources (so-called cake-cutting), and indivisible resources. We also describe our results for indivisible resources with and without monetary transfers.

9.2 Basic Problems and Algorithms for Mobility Society

In this section, we introduce the fundamental optimization problems: the shortest path and the traveling salesman problems. We then outline their differences from the viewpoint of the computational complexity. Additionally, we briefly describe a series of studies on developing efficient algorithms.

The primary task of an automotive navigation system is to determine a shortest route from an individual's current location to a given destination. This problem is modeled as the *shortest path problem*, which is described as follows. Let $G = (V, E)$ be a directed graph with a vertex set V and an edge set $E \subseteq V \times V$. The vertices correspond to road intersections, and the edges correspond to road segments. Each edge $(u, v) \in E$ has a positive real number $\ell(u, v)$ called *weight*, which represents the length of the corresponding road segment, or the cost to travel from u to v on the road segment. The shortest path problem is given an edge-weighted graph and two vertices, s and t, called the source and sink, respectively, to find a path from the source s to the sink t such that the sum of the weights in the path is minimized. In general, we may allow for negative edge weights in the shortest path problem. However, in this chapter, we only allow nonnegative edge weights because the edge weights in many mobility-related applications often correspond to length, time, or nonnegative costs.

The *traveling salesman problem* (TSP) is the problem of determining a shortest tour that visits all the vertices exactly once for a given edge-weighted graph. The TSP has been extensively studied since it has several applications, such as planning, logistics, and microchip manufacturing. As a logistical example, let us consider a single van driver who delivers packages from a depot station to customers' homes. The driver needs a shortest tour that visits all the homes and returns to the depot. In this case, we have a complete graph $G = (V, V \times V)$ with the vertices V corresponding to the depot and homes, and edge weights representing the travel time or cost between the depot and homes, corresponding to the edges' endpoints. Then, the problem of finding a shortest tour is formulated as the TSP.

The shortest path problem admits an algorithm that runs in polynomial time [17]. Theoretically, such an algorithm is considered to be efficient. Thus, constructing polynomial-time algorithms for problems is one of the ultimate goals in the field of optimization and computational complexity. On the other hand, the TSP is known to be NP-hard [32], which implies that the TSP is unlikely to be solved in polynomial time.

Meanwhile, many NP-hard problems, including the TSP, arise in the real world. Hence, considerable effort has been made to develop fast algorithms for NP-hard problems from both theoretical and practical viewpoints. Owing to the NP-hardness, we need to compromise the running time of the algorithms and/or the optimality of their outcomes, which leads to two main approaches: finding an exact optimal solution as quickly as possible and finding a sufficiently good solution quickly.

9.2.1 Approaches for Hard Problems

Most combinatorial optimization problems can be trivially solved by a brute-force search, that is, enumerating and checking all possible solutions. Thus, determining how to yield a performance better than a brute-force search is an important research topic. Theoretical research aimed at answering this question began in the 1960s. For example, the dynamic programming algorithms proposed by Bellman [8] and Held and Karp [27] in 1962 were used to solve the TSP with n vertices in $O(2^n n^2)$ time. This bound is considerably faster than a simple brute-force search, which requires $O(n \cdot n!)$ time. Currently, these algorithms are referred to as the *exact exponential algorithms*. For further details, see the book of Fomin and Kratsch [20]. *Parameterized complexity theory*, a similar area of research introduced by Downey and

Fellows [18], aims to provide a refined analysis of NP-hard problems by using their parameters. In particular, this theory focuses on the existence of algorithms whose running times are bounded by a polynomial in the input length and a function for the parameters. An overview of this approach has been presented by Flum and Grohe [19] and Niedermeier [43]. From a practical viewpoint, many useful methods are based on logic such as satisfiability (SAT) solvers, and mathematical programming such as mixed integer programming (MIP) solvers. There are several mixed integer programming formulations of the TSP. Practical MIP solvers enable us to obtain an (exact) optimal solution for the medium-scale TSP.

In real-world applications, it is often adequate to determine a sufficiently good solution quickly instead of taking much time to find an optimal solution. Therefore, researchers have developed approximation algorithms and heuristics to solve real-world problems. Here, an *approximation algorithm* is referred to as a polynomial-time algorithm with a performance guarantee. For a problem instance I of a minimization problem, let $\mathrm{ALG}(I)$ denote the objective value obtained by an algorithm ALG, and let $\mathrm{OPT}(I)$ denote the optimal value of the instance I. For ρ (≥ 1), algorithm ALG is said to be a *ρ-approximation* if $\mathrm{ALG}(I) \leq \rho \cdot \mathrm{OPT}(I)$ holds for any problem instance I. Approximation algorithms for various difficult problems have been extensively studied since the 1990s.

Unfortunately, the general TSP is known to be hard even to approximate, unless $\mathrm{P} = \mathrm{NP}$. This means that, for any polynomial-time algorithm, there exists a problem instance in which the output value of the algorithm is infinitely large compared with the optimal value, unless $\mathrm{P} = \mathrm{NP}$. On the other hand, the metric TSP, which is an important special case of the TSP, admits approximation algorithms. Let $\mathbb{R}_+$ be the set of nonnegative real numbers. The TSP is called *metric* if the vertex set V and the edge weight $\ell\colon V \times V \to \mathbb{R}_+$ form a *metric space*, that is, ℓ satisfies three conditions: (1) non-degeneracy $\ell(u, v) = 0$ if and only if $u = v$, (2) the symmetry $\ell(u, v) = \ell(v, u)$ for any $u, v \in V$, and (3) the triangle inequality $\ell(u, v) \leq \ell(u, w) + \ell(w, v)$ for any $u, v, w \in V$. A celebrated result of Christofides [12] is a 1.5-approximation algorithm for the metric TSP. This algorithm finds a tour with a length no more than 1.5 times the minimum tour length. A typical example of the metric TSP is when the vertices are assumed to be embedded in the Euclidean space, and the edge weights are defined as the Euclidean distances $\ell(u, v) = \|u - v\|_2$. This problem is called the *Euclidean* TSP, and is naturally occurring from real-world applications such as microchip manufacturing. The Euclidean TSP further admits a polynomial-time approximation scheme [3]. Specifically, for each fixed positive number $\varepsilon > 0$, we can find a $(1 + \varepsilon)$-approximate solution in polynomial time. The Christofides' 1.5-approximation algorithm has been the best approximation algorithm for the general metric TSP for several decades. Recently, improved algorithms for the general metric TSP have also been developed [30, 31]. In addition, substantial progress has been made in regard to algorithms for a special case, called graphic TSP [21, 40, 42, 45].

Simultaneously, considerable effort has been devoted to developing better heuristics from a practical viewpoint. A well-known heuristic for the TSP is k-opt. The basic idea of k-opt is to repeatedly remove k disjoint edges and reassemble the fragments into a different tour. Generalizing this, the Lin–Kernighan method [35] repeats the

k-opt procedure by varying k. This is one of the core techniques to construct better heuristics for the TSP when the edge weights are symmetric. Some studies have also applied general heuristics, such as genetic algorithms and simulated annealing, to the TSP.

Finally, we note that a polynomial-time algorithm does not necessarily run quickly in practice, particularly when the problem is large. Therefore, even though the shortest path problem is solvable in polynomial time, a lot of additional effort has been made to find a short path quickly. For example, Dijkstra's algorithm [17], which is a well-known polynomial-time algorithm for the shortest path problem, is impractical for a large given graph (e.g., a huge load network or SNS network). This is because this algorithm essentially determines a shortest path tree, which consists of shortest paths from the source to all the other vertices. To overcome this drawback, the A* algorithm [26] was proposed to determine a shortest path from the source to the sink, rather than a shortest path tree, by incorporating heuristics into Dijkstra's algorithm. Approximation algorithms for the shortest path problem have also attracted attention [1, 7].

9.2.2 Reallocation Scheduling

In this subsection, we present the results for reallocation scheduling, which are obtained in the project. Reallocation scheduling is a fundamental problem in various fields, such as supply chain management, logistics, and transportation science. Many reallocation models have been studied from both theoretical and practical perspectives. Reallocation models are categorized based on three aspects: (i) reallocation cost, (ii) fungibility of products, and (iii) parallel/sequential execution.

For example, let us consider the *dial-a-ride* problem, which involves designing schedules for vehicles to handle customer requests. Specifically, we consider a company that provides delivery services. This company dispatches identical delivery vehicles to pick up and drop off packages at the requested locations. Each request involves carrying a package between two specified locations. To process a request, a vehicle must first move to the pickup location, and then the vehicle transports the package to its drop-off location. The vehicle completes the process by handing over the package. The problem is to find a schedule that minimizes some objective, for example, the makespan or sum of the completion times of the vehicles. Here, the *makespan* is the time at which all vehicles return to the depot after completing all requests. This problem can be regarded as a vehicle routing problem for reallocation [15, 25, 28]. In the dial-a-ride problem, (i) the reallocation cost depends on the route; (ii) customers are not fungible; and (iii) reallocation is performed sequentially for each vehicle. Another example is the bike-share rebalancing problem, which involves scheduling trucks to redistribute shared bikes with the objective of minimizing the total cost [16]. In this problem, conditions (i) and (iii) for the dial-a-ride problem also apply: the reallocation cost depends on the route, and reallocation is performed sequentially for each truck. However, condition (ii) does not apply: shared bikes are fungible. That is, we can fulfill the desired bike distribution without distinguishing between bikes. In both of these problems, costs arise from the transportation of the delivery vehicles rather than the reallocation of the products.

In the project, Ishii et al. [29] considered reallocation scheduling in which (i) the cost (i.e., transit time) of reallocating a product is given in advance; (ii) products are not fungible; and (iii) reallocations are performed in a parallel manner. We refer to this problem as the reallocation problem. For example, suppose that several warehouses store many products (or items). Some products are already stored in their designated warehouses, whereas others are stored in temporary warehouses; the latter must be reallocated to their designated warehouses. To reallocate a product p, a certain amount of time is required, which is called the transit time of p. Each product also has a size, and each warehouse has three types of capacity: (a) the capacity of the warehouse itself, (b) the carry-in size capacity, and (c) the carry-out size capacity. The capacity of type (a) restricts the total number of products that can be stored in a warehouse at any moment, while (b) and (c) restrict the total size of products that are simultaneously carried in and out, respectively. For this setting, we provided an algorithm to find a reallocation schedule with the minimum completion time.

The project investigated the computational complexity of the reallocation problem under various scenarios and obtained the following results.

Theorem 9.1 (Ishii et al. [29]) *The reallocation problem with uniform product sizes and uniform transit times can be solved in* $\mathrm{O}(mn \log m)$ *time, where* m *and* n *denote the number of products and warehouses, respectively.*

Theorem 9.2 (Ishii et al. [29]) *The reallocation problem is NP-hard in general.*

The paper [29] also provided approximation algorithms for the reallocation problem under several scenarios that arise in real applications.

9.3 Online Optimization for Mobility Society

In online problems, the complete data are not known in advance, but are revealed piece by piece. Decisions must be made based only on the currently nown information, and not on future information. Online problems have been actively studied in the field of theoretical computer science since the seminal paper by Sleator and Tarjan in 1985 [46]. The authors suggested comparing an online algorithm with an optimal offline algorithm to measure the performance of the online algorithm in a manner similar to an approximation ratio. For an input sequence σ, let $\mathrm{ALG}(\sigma)$ and $\mathrm{OPT}(\sigma)$, respectively, denote the costs obtained by an online algorithm ALG and the optimal offline algorithm OPT. For ρ (≥ 1), algorithm ALG is said to be (asymptotically) *ρ-competitive* if a nonnegative constant α exists such that

$$\mathrm{ALG}(\sigma) \leq \rho \cdot \mathrm{OPT}(\sigma) + \alpha$$

for any input sequence σ.

Sleator and Tarjan [46] studied the *paging problem*, which is an important problem in implementing computer operating systems. For example, consider a two-level memory system consisting of primary and secondary memories. A primary memory is typically small, while a secondary one is large. Suppose that we have a sequence of requests, each of which specifies a page in the memory system. A request is served if the corresponding page is in the primary memory. If a page is not in the primary memory, a *page fault* occurs. Then, some pages must be removed from the primary memory, and the requested page must be loaded from the secondary memory into the primary memory. The aim is to minimize the number of page faults incurred in the request sequence. Sleator and Tarjan [46] proved that any algorithm for the paging problem is at least k-competitive, and that algorithms so-called *least recently used* and *first-in first-out* are k-competitive.

The *k-server problem* proposed by Manasse et al. [39] is a generalization of the paging problem and is one of the most extensively studied online problems. As an example, suppose that we have k mobile servers located in a metric space. In this problem, we are given a sequence of service requests, each of which specifies a point to visit in the space. When a request arrives, we must immediately select a server to move to the requested point without knowing future requests. The next request is known when the current request is completed. The aim is to determine how to assign servers to requests to minimize the total moving distance moved by the servers. Manasse et al. [39] conjectured that a k-competitive algorithm exists for the k-server problem. Koutsoupias and Papadimitriou [34] proved that the *work function algorithm* has a competitive ratio of at most $2k - 1$ for any metric. Despite many efforts, the conjecture is still open.

Coester and Koutsoupias [13] studied the *online k-taxi problem*, which is a generalization of the k-server problem. In this problem, k taxis serve a sequence of requests in a metric space; a request consists of two points s and t, representing a passenger who wants to be carried by a taxi from s to t.

The online k-server and k-taxi problems assume that a new request arrives after the current request has been processed. To handle a situation in which each request arrives at a certain time, we require a real-time model. The *online k-TSP* problem is a real-time version of the k-server problem, wherein each request arrives at a release time, and the servers can move at a unit speed. When the requests are provided in advance, and there is only one server, the problem coincides with the TSP. The *online k-dial-a-ride* problem is a real-time version of the k-taxi problem. This is also a generalization of the online k-TSP. The goal of this problem is to minimize the makespan. Ascheuer et al. [4] proposed the *SMARTSTART* algorithm, which achieves the best possible competitive ratio of 2.

Goko et al. [23] introduce a scheduling problem called the *online $(\boldsymbol{M}, k)$-scheduling problem*, which generalizes the online k-dial-a-ride problem. In the problem, k machines move in the metric space $\boldsymbol{M}$. For example, let us consider a company that provides on-demand delivery services. Similar to the online k-dial-a-ride problem, we need to design schedules for vehicles to process all requests such that the makespan is minimized. However, unlike the online k-dial-a-ride problem, we do not know the destination state of a request until we begin to process it. Furthermore, the time required to process a request is not known until the end of the process, because the time required for pickup at the source and handing over at the destination are also included. Here, we assume that each vehicle can only carry one package at a time.

In the paper [23], Goko et al. provided the following result.

Theorem 9.3 (Goko et al. [23]) *The online* $(\boldsymbol{M}, k)$*-scheduling problem is* $\Theta(\frac{\log k}{\log \log k})$*-competitive.*

This theorem means that there exists an $O(\log k/ \log \log k)$-competitive online algorithm and every online algorithm is $\Omega(\log k/ \log \log k)$-competitive. This result is obtained in the following two steps. First, to address the unknown release times, we provide frameworks for producing a $\min\{2\rho + 1/2,\ \rho + 2\}$-competitive algorithm using a ρ-competitive algorithm for the *basic* online $(\boldsymbol{M}, k)$-scheduling problem, in which all jobs are released at time 0. Then, to address the unknown destination states and processing times, we construct an $O(\log k/ \log \log k)$-competitive algorithm for the basic problem. We also improved the competitive ratios of the online $(\boldsymbol{M}, k)$-scheduling problem for certain special cases.

9.4 Mechanism Design for Mobility Society

In traditional optimization problems, only one agent makes decisions. However, real-world problems often involve multiple agents, each of whom aims to optimize their own gains. In this scenario, an outcome should be satisfactory for everyone. The preferable outcome is formulated as a solution that satisfies concepts such as equilibrium, stability, envy-freeness, proportionality, and popularity. An algorithm that achieves a desirable outcome is essential for our livelihood. For example, when several passengers ride in the same taxi, they typically wish to split the taxi fare fairly. In terms of mechanism design, one way to achieve this end is to calculate the payment for each passenger based on the *Shapley value*, which is a basic solution concept in cooperative game theory. A Shapley value of a passenger is, roughly speaking, the passenger's average marginal contribution to the taxi fare of any subset of other passengers.

In the joint project, we have studied stable matchings [24, 38] and fair division [2, 22, 33, 37]. In the remainder of this section, we describe the results for fair division in [33, 37]. Specifically, we define the solution concepts of fair division, such as envy-freeness and proportionality in the fair division. Then we outline the related results from the joint project.

Fair division is the problem of dividing a set of resources among several agents in such a way that agents receive their due share. This problem arises in various real-world settings, such as the division of inheritance, electronic frequency allocation, and airport traffic management. Cake-cutting is a fundamental fair division problem, which has been studied in economics and mathematics since 1940s. Let $N = \{1, 2, \ldots, n\}$ be a set of n agents. A cake C is a heterogeneous divisible good;

let C be represented by the interval $[0, 1]$. Let $\mathfrak{C}$ denote the Borel subsets of C.[1] Each agent $i \in N$ has a *valuation measure* $v_i : \mathfrak{C} \to \mathbb{R}_+$. An n-tuple $(C_1, \dots, C_n) \in \mathfrak{C}^n$ is called a *partition* of C if $\bigcup_{i \in N} C_i = C$ and $C_i \cap C_j = \emptyset$ for any distinct $i, j \in N$. For a partition $(C_1, \dots, C_n)$ of C, let agent $i \in N$ receive C_i. A partition $(C_1, \dots, C_n)$ of C is called *envy-free* (*EF*) if $v_i(C_i) \geq v_i(C_j)$ for any $i, j \in N$. This means that every agent i has no incentive to exchange the agent's own piece C_i with another C_j.

In the literature, we assume that each valuation measure v_i $(i \in N)$ satisfies the following natural properties:

1. Normalization: $v_i(C) = 1$ and $v_i(\emptyset) = 0$.
2. Non-atomicity: $v_i(\{x\}) = 0$ for every single point $x \in C$.
3. σ-additivity: $v_i(\bigcup_{k=1}^{\infty} X_k) = \sum_{k=1}^{\infty} v_i(X_k)$ for every sequence of disjoint sets $X_1, X_2, \dots \subseteq \mathfrak{C}$.

For such valuations, an EF partition of the cake always exists [47]. For example, if $n = 2$, the *cut-and-choose* (also called *divide-and-choose*) protocol can find an EF partition. In this protocol, one agent, called the *cutter*, cuts the cake into two pieces, the other agent, called the *chooser*, chooses one of the pieces, and the cutter receives the remaining piece.

Proportionality is another fairness criterion. A partition $(C_1, \dots, C_n)$ of C is called *proportional* (*PROP*) if $v_i(C_i) \geq 1/n \cdot v_i(C)$ for any $i \in N$. This means that every agent receives at least the agent's due share according to the agent's own valuation measure. By definition, envy-freeness implies proportionality; hence, a proportional partition always exists if valuation measures satisfy the properties above.

Recently, the fair division of a set of indivisible goods, such as cars and houses, has been studied extensively in computer science. This problem can be modeled as follows. Let $M = \{1, 2, \dots, m\}$ be a set of m indivisible goods. Each agent $i \in N$ has a *valuation function* $v_i : 2^M \to \mathbb{R}_+$ with $v_i(\emptyset) = 0$. We assume that the valuation functions are *monotone*: $v_i(X) \leq v_i(Y)$ for any $X \subseteq Y \subseteq M$. An allocation of indivisible goods $A = (A_1, \dots, A_n)$ is a partition of M into n bundles, that is, it satisfies that $\bigcup_{i \in N} A_i = M$ and $A_i \cap A_j = \emptyset$ for any distinct i and j in N.

Unlike the cake-cutting problem, an EF or PROP partition of indivisible goods may not exist. As an extreme example, if there is only one good which is demanded by every agent, it is given to a single agent; however, the other agents have envy. Specifically, any partition is not PROP, and hence, not EF.

Thus, the following relaxed criteria of EF have been studied in the literature by Budish [9] and Caragiannis et al. [10].

- EFX [10]: an allocation A is called *envy-free up to any good* (*EFX*) if for any agents $i, j \in N$, we have $v_i(A_i) \geq v_i(A_j \setminus \{e\})$ for every $e \in A_j$.
- EF1 [9]: an allocation A is called *envy-free up to one good* (*EF1*) if for any agents $i, j \in N$, either $v_i(A_i) \geq v_i(A_j)$ or there exists an $e \in A_j$ such that $v_i(A_i) \geq v_i(A_j \setminus \{e\})$.

[1] The Borel subsets of C are the sets formed by taking all possible countable unions, intersections, and complements of open sets within C.

Any set of indivisible goods admits an EF1 allocation [36]. It is known that there exists an EFX allocation for 2 agents [44], while the existence is open for at least 3 agents. When each agent has an additive valuation, i.e., a valuation function v_i of each agent i satisfies $v_i(X) = \sum_{e \in X} v_i(\{e\})$ for any subset $X \subseteq M$, it is known that an EFX allocation always exists for 3 agents [11], while its existence is open for at least 4 agents.

In the joint project, Mahara [37] revealed a new class of fair division instances that admit EFX allocations.

Theorem 9.4 (Mahara [37]) *An EFX allocation exists when each agent has one of two valuation functions.*

Similar to the relaxation of EF to EF1 and EFX, two relaxed concepts of PROP have been proposed by Conitzer et al. [14] and Moulin [41].

- PROPx [5, 41]: an allocation A is called *proportional up to the least valued good* (*PROPx*) if for any agent $i \in N$, either $v_i(A_i) \geq 1/n \cdot v_i(M)$ or for every $e \in M \setminus A_i$, we have $v_i(A_i \cup \{e\}) \geq 1/n \cdot v_i(M)$.[2]
- PROP1 [14]: an allocation A is called *proportional up to the highest valued good* (*PROP1*) if for any agent $i \in N$, either $v_i(A_i) \geq 1/n \cdot v_i(M)$ or there exists an $e \in M \setminus A_i$ such that $v_i(A_i \cup \{e\}) \geq 1/n \cdot v_i(M)$.

A PROPx allocation may not exist [5, 41], while a PROP1 allocation always exists [14].

In the joint project, a new concept related to PROP1 and PROPx was proposed for additive valuations [33]. An allocation A is called *proportionality up to the least valued good on average* (*PROPavg*) if it holds that

$$v_i(A_i) + \frac{1}{n-1} \sum_{k \in N \setminus \{i\}} \operatorname*{Min}_{e \in A_k} v_i(\{e\}) \geq \frac{1}{n} v_i(M), \tag{9.1}$$

where we define $\mathrm{Min}(S) = \min(S)$ for $S \neq \emptyset$ and $\mathrm{Min}(\emptyset) = 0$. In other words, agent i receives a set of goods that she values as at least $1/n$ fraction of the total value of all goods, minus the average minimum value of the set of goods received by any other agent. Note that for additive valuations, PROPavg is a concept that lies between PROP1 and PROPx: any PROPx allocation is PROPavg, and any PROPavg allocation is PROP1. Moreover, PROPavg is a relaxation of *Avg-EFX*, which is introduced by Baklanov et al. [6]. The notion of Avg-EFX is obtained by replacing $n-1$ with n on the left-hand-side of Eq. (9.1):

$$v_i(A_i) + \frac{1}{n} \sum_{k \in N \setminus \{i\}} \operatorname*{Min}_{e \in A_k} v_i(\{e\}) \geq \frac{1}{n} v_i(M).$$

[2] The name of PROPx is proposed by Aziz et al. [5].

In the paper [33] of the project, Kobayashi and Mahara demonstrated that a PROPavg allocation always exists; however, the existence of Avg-EFX allocations remains open [6].

Theorem 9.5 (Kobayashi and Mahara [33]) *A PROPavg allocation always exists when each agent has an additive valuation.*

The joint project also studied EF allocations with subsidies [22] and fair ride allocations based on the Shapley value [2].

Acknowledgements We would like to thank Ayumi Igarashi, Toshimasa Ishii, Jun Kawahara, Yusuke Kobayashi, Ryoga Mahara, and Hirotaka Ono for helping us to write this manuscript. This work was supported by the joint project between Kyoto University and Toyota Motor Corporation, titled "Advanced Mathematical Science for Mobility Society."

References

1. G. Aggarwal, S. Gollapudi, Raghavender, A.K. Sinop, Sketch-based algorithms for approximate shortest paths in road networks, in *Proceedings of the Web Conference 2021* (2021), pp. 3918–3929
2. Y. Amano, A. Igarashi, Y. Kawase, K. Makino, H. Ono, Fair ride allocation on a line, in *Proceedings of the 15th International Symposium on Algorithmic Game Theory* (2022), pp. 421–435
3. S. Arora, Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. J. ACM **45**(5), 753–782 (1998)
4. N. Ascheuer, S.O. Krumke, J. Rambau, Online dial-a-ride problems: minimizing the completion time, in *Proceedings of the 17th Annual Symposium on Theoretical Aspects of Computer Science* (2000), pp. 639–650
5. H. Aziz, H. Moulin, F. Sandomirskiy, A polynomial-time algorithm for computing a Pareto optimal and almost proportional allocation. Oper. Res. Lett. **48**(5), 573–578 (2020)
6. A. Baklanov, P. Garimidi, V. Gkatzelis, D. Schoepflin, Achieving proportionality up to the maximin item with indivisible goods, in *Proceedings of the 35th AAAI Conference on Artificial Intelligence* (2021), pp. 5143–5150
7. R. Becker, S. Forster, A. Karrenbauer, C. Lenzen, Near-optimal approximate shortest paths and transshipment in distributed and streaming models. SIAM J. Comput. **50**(3), 815–856 (2021)
8. R. Bellman, Dynamic programming treatment of the travelling salesman problem. J. ACM **9**(1), 61–63 (1962)
9. E. Budish, The combinatorial assignment problem: approximate competitive equilibrium from equal incomes. J. Political Econ. **119**(6), 1061–1103 (2011)
10. I. Caragiannis, D. Kurokawa, H. Moulin, A.D. Procaccia, N. Shah, J. Wang, The unreasonable fairness of maximum Nash welfare. ACM Trans. Econ. Comput. **7**(3), 1–32 (2019)
11. B.R. Chaudhury, J. Garg, K. Mehlhorn, EFX exists for three agents, in *Proceedings of the 21st ACM Conference on Economics and Computation* (2020), pp. 1–19
12. N. Christofides, Worst-case analysis of a new heuristic for the traveling salesman problem. Report 388, Graduate School of Industrial Administration, Carnegie Mellon University (1976)
13. C. Coester, E. Koutsoupias, The online k-taxi problem, in *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing* (2019), pp. 1136–1147
14. V. Conitzer, R. Freeman, N. Shah, Fair public decision making, in *Proceedings of the 18th ACM Conference on Economics and Computation* (2017), pp. 629–646

15. J.F. Cordeau, G. Laporte, The dial-a-ride problem: models and algorithms. Ann. Oper. Res. **153**(1), 29–46 (2007)
16. M. Dell'Amico, E. Hadjicostantinou, M. Iori, S. Novellani, The bike sharing rebalancing problem: mathematical formulations and benchmark instances. Omega **45**, 7–19 (2014)
17. E.W. Dijkstra, A note on two problems in connexion with graphs. Numer. Math. **1**(1), 269–271 (1959)
18. R.G. Downey, M.R. Fellows, *Parameterized Complexity* (Springer, New York, 1999)
19. J. Flum, M. Grohe, *Parameterized Complexity Theory* (Springer, Berlin, Heidelberg, 2006)
20. F.V. Fomin, D. Kratsch, *Exact Exponential Algorithms* (Springer, Berlin, Heidelberg, 2010)
21. S.O. Gharan, A. Saberi, M. Singh, A randomized rounding approach to the traveling salesman problem, in *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science* (2011), pp. 550–559
22. H. Goko, A. Igarashi, Y. Kawase, K. Makino, H. Sumita, A. Tamura, Y. Yokoi, M. Yokoo, Fair and truthful mechanism with limited subsidy, in *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems* (2022), pp. 534–542
23. H. Goko, A. Kawamura, Y. Kawase, K. Makino, H. Sumita, Online scheduling on identical machines with a metric state space, in *Proceedings of the 39th International Symposium on Theoretical Aspects of Computer Science* (2022), pp. 32:1–32:21
24. H. Goko, K. Makino, S. Miyazaki, Y. Yokoi, Maximally satisfying lower quotas in the hospitals/residents problem with ties, in *Proceedings of the 39th International Symposium on Theoretical Aspects of Computer Science* (2022), pp. 31:1–31:20
25. I.L. Gørtz, V. Nagarajan, R. Ravi, Minimum makespan multi-vehicle dial-a-ride. ACM Trans. Algorithms **11**(3), 1–29 (2015)
26. P.E. Hart, N.J. Nilsson, B. Raphael, A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Syst. Man Cybern. Syst. **4**(2), 100–107 (1968)
27. M. Held, R.M. Karp, A dynamic programming approach to sequencing problems. J. Soc. Ind. Appl. Math. **10**(1), 196–210 (1962)
28. S.C. Ho, W.Y. Szeto, Y.-H. Kuo, J.M.Y. Leung, M. Petering, T.W.H. Tou, A survey of dial-a-ride problems: Literature review and recent developments. Transp. Res. Part B: Methodol. **111**, 395–421 (2018)
29. T. Ishii, J. Kawahara, K. Makino, H. Ono, Reallocation problems with minimum completion time, in *Proceedings of the 28th International Computing and Combinatorics Conference* (2022), pp. 292–304
30. A.R. Karlin, N. Klein, S.O. Gharan, A (slightly) improved approximation algorithm for metric TSP, in *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing* (2021), pp. 32–45
31. A.R. Karlin, N. Klein, S.O. Gharan, A (slightly) improved deterministic approximation algorithm for metric TSP (2022). arXiv:2212.06296
32. R.M. Karp, Reducibility among combinatorial problems, in *Complexity of Computer Computations* (Springer, US, Boston, MA, 1972), pp.85–103
33. Y. Kobayashi, R. Mahara, Proportional allocation of indivisible goods up to the least valued good on average, in *Proceedings of the 33rd International Symposium on Algorithms and Computation* (2022), pp. 55:1–55:13
34. E. Koutsoupias, C.H. Papadimitriou, On the k-server conjecture. J. ACM **42**(5), 971–983 (1995)
35. S. Lin, B.W. Kernighan, An effective heuristic algorithm for the traveling-salesman problem. Oper. Res. **21**(2), 498–516 (1973)
36. R.J. Lipton, E. Markakis, E. Mossel, A. Saberi, On approximately fair allocations of indivisible goods, in *Proceedings of the 5th ACM Conference on Electronic Commerce* (2004), pp. 125–131
37. R. Mahara, Extension of additive valuations to general valuations on the existence of EFX, in *Proceedings of the 29th Annual European Symposium on Algorithms* (2021), pp. 66:1–66:15
38. K. Makino, S. Miyazaki, Y. Yokoi, Incomplete list setting of the hospitals/residents problem with maximally satisfying lower quotas, in *Proceedings of the 15th International Symposium on Algorithmic Game Theory* (2022), pp. 544–561

39. M.S. Manasse, L.A. Mcgeoch, D.D. Sleator, Competitive algorithms for server problems. J. Algorithms **11**, 208–230 (1990)
40. T. Mömke, O. Svensson, Removing and adding edges for the traveling salesman problem. J. ACM **63**(1), 1–28 (2016)
41. H. Moulin, Fair division in the internet age. Annu. Rev. Econ. **11**, 407–441 (2019)
42. M. Mucha, $\frac{13}{9}$-approximation for graphic TSP. Theory Comput. Syst. **55**(4), 640–657 (2014)
43. R. Niedermeier, *Invitation to Fixed-Parameter Algorithms* (Oxford University Press, 2006)
44. B. Plaut, T. Roughgarden, Almost envy-freeness with general valuations. SIAM J. Discrete Math. **34**(2), 1039–1068 (2020)
45. A. Sebő, J. Vygen, Shorter tours by nicer ears: 7/5-approximation for the graph-TSP, 3/2 for the path version, and 4/3 for two-edge-connected subgraphs. Combinatorica **34**(5), 597–629 (2014)
46. D.D. Sleator, R.E. Tarjan, Amortized efficiency of list update and paging rules. Commun. ACM **28**(2), 202–208 (1985)
47. H.R. Varian, Equity, envy and efficiency. J. Econ. Theory **9**, 63–91 (1974)
48. Wikipedia. Automotive navigation system—Wikipedia, the free encyclopedia (2023). http://en.wikipedia.org/w/index.php?title=Automotive%20navigation%20system&oldid=1142632441

Chapter 10
Mechanism Design for Mobility

Tsubasa Harada, Toshiya Itoh, Shigeo Matsubara, Shuichi Miyazaki, and Makoto Yokoo

10.1 Introduction

This chapter discusses mechanism design for mobility. With goals such as reducing carbon dioxide emissions and promoting resource conservation being imposed globally, car sharing becomes increasingly important. On the other hand, with various services coming online, we are now in an era where the needs of individual users can be met more flexibly. The same is true for car sharing and other transportation services. Uber, for example, is no different from a conventional cab in that it transports people to their destinations. However, it offers added value to both drivers and users, such as improved user experience in that payment is completed on a smartphone application and a driver evaluation system is introduced, as well as a response to supply-demand mismatches by adjusting the fare.

Here, because the physical movement of people and vehicles is the essence of mobility services, mobility services have different properties from other services that are completed online. In other words, an important issue of developing mobility

T. Harada · T. Itoh
Tokyo Institute of Technology, 2-12-1 Ookayama, Meguro-ku, Tokyo 152-8550, Japan
e-mail: harada.t.ak@m.titech.ac.jp

T. Itoh
e-mail: titoh@c.titech.ac.jp

S. Matsubara (✉)
Osaka University, 1-3 Machikaneyama-cho, Toyonaka, Osaka 560-8531, Japan
e-mail: matsubara@sigmath.es.osaka-u.ac.jp

S. Miyazaki
University of Hyogo, 8-2-1 Gakuennishi-machi, Nishi-ku, Kobe, Hyogo 651-2197, Japan
e-mail: shuichi@sis.u-hyogo.ac.jp

M. Yokoo
Kyushu University, 744 Motooka, Nishi-ku, Fukuoka 819-0395, Japan
e-mail: yokoo@inf.kyushu-u.ac.jp

K. Ikeda et al. (eds.), *Advanced Mathematical Science for Mobility Society*,
https://doi.org/10.1007/978-981-99-9772-5_10

technology is how to integrate physical constraints with digital services' flexibility so that it meets the needs of many potential users and is accepted by them. In examining this question, it is helpful to introduce the concept of mechanism design.

Mechanism design is a branch of economics that aims to design rules that enable autonomous/decentralized realization of goals in areas such as resource allocation and social decision-making. Mechanism design studies are not limited to basic research but have also been very successful in real-world applications, including spectrum auctions, school choice mechanisms in public schools, and kidney donor matching.

In recent years, computational approaches to mechanism design have become popular and have developed significantly as an interdisciplinary area with artificial intelligence and theoretical computer science [35]. In particular, the contributions of computer science have been remarkable, such as the development of algorithms for kidney donor matching that can obtain a solution in a reasonable amount of time, even when the number of donors is large [1]

This chapter presents two topics in developing basic research on matching theories and one topic in applying the matching theory to car sharing. Section 10.2 discusses online facility assignment problems on a line. One challenge for developing basic research toward mobility applications is its online nature [33]. For example, ride-hailing, one form of car sharing, involves passengers hiring a driver to take them to their destinations. The operator of the ride-hailing system must decide which car to assign to each usage request without knowing in advance exactly when and where future requests will occur. Here, it is important to understand what algorithm effectively achieves social goals such as minimizing total travel distance.

Section 10.3 discusses two-sided matching with flexible quotas: student-project-resource matching-allocation problem.[1] Another challenge for developing basic research toward mobility applications is dealing with preferences. Here, in the design of vehicle-sharing systems, the medium-term issues include how to provide proper incentives to users, and the long-term issues include how to design parking facilities [12]. Some studies have introduced the preference argument into mobility design. However, the previous studies have discussed the matching problem for the medium-term issues and the resource allocation problem for the long-term issues separately. Integrating these two problems will produce more desirable outcomes for individuals and society.

On the application side, Sect. 10.4 shows how matching theory can be applied to station-based one-way car sharing. So far, the target of applying matching theory is often ride-sharing in that drivers and riders have preferences for each other [47]. However, station-based car sharing is also widely deployed in which multiple drivers share a single car, and a supply-demand mismatch problem still needs to be solved. Therefore, we focus on negotiations among drivers to solve a supply-demand mismatch problem, which can be naturally expressed as preference matching.

[1] The term student is used because the many-to-one matching problem is often discussed in the context of the school choice problem. Here, it can be considered as representing passengers or vehicles.

Note that the studies in this chapter aim to build the essential elements of the next-generation car-sharing system, but they can also be applied to other mobility applications, such as controlling logistics systems.

10.2 Online Facility Assignment Problems on a Line

In this section, we summarize the results of the competitive analysis for the online facility assignment problem on a line, which appear in [17, 20].

10.2.1 Background and Known Results

Online optimization (profit maximization or cost minimization) problems are real-time computations, in which a sequence of requests is an input, each request is given to an online algorithm one by one, and an online algorithm must decide how to deal with the current request before the next one is given. Once the decision is made for a request, it cannot be changed later. In general, the efficiency of online algorithms is measured by competitive analysis, which is initiated by Sleator and Tarjan [44]. Informally, we say that an online algorithm ALG is *c-competitive* (or the *competitive ratio* of ALG is at most c) if the cost of the solution output by ALG is at most c times worse than the optimal cost (the formal definition will be given in Sect. 10.2.2.2).

The *online metric matching problem* is initiated independently by Kalyanasundaram and Pruhs [22] and Khuller et al. [26] as an online variant of the minimum cost bipartite matching problem, and is formulated as follows: k servers are located on a given metric space and k requests (on the same metric space) are given one by one in an online manner. The task of an online algorithm is to match each request immediately with one of the k servers. The cost of matching a request with a server is determined by the distance between them. The goal of the problem is to minimize the sum of the costs of matching k requests to k distinct servers. For this problem, Kalyanasundaram and Pruhs [22] and Khuller et al. [26] presented a deterministic online algorithm, which is called *Permutation* [22], and showed that it is $(2k-1)$-competitive and best possible.

Later, Kalyanasundaram and Pruhs [23] restricted the underlying metric space to a line and introduced a problem referred to as the online matching problem on a line. For this restricted problem, the best upper bound on the competitive ratio is $O(\log k)$ [34, 39], which is achieved by the ROBUST- MATCHING algorithm [38]. While the best lower bound on the competitive ratio has been 9.001 for a long time [10], Peserico and Scquizzato [37] drastically improved it to $\Omega(\sqrt{\log k})$.

As a variant of the online metric matching problem, Ahmed et al. [2] formulated the *online facility assignment* problem as follows: There exist k servers located equidistantly on a line and each request appears (one by one) on the line. Each server has a *capacity*, which corresponds to the possible number of requests that can be matched to the server. Ahmed et al. [2] showed (with rough proofs) that the greedy algorithm is $4k$-competitive. They also proposed an *Optimal-fill* algorithm and showed that the algorithm is k-competitive for any $k > 2$.

10.2.2 Preliminaries

10.2.2.1 Problem Definition

Let (X, d) be a metric space, where X is a (possibly infinite) set of points and $d : X \times X \to \mathbb{R}$ is a distance function. We use $S = \{s_1, \dots, s_k\}$ to denote the set of k servers and use $\sigma = r_1 \cdots r_n$ to denote a request sequence, where each r_i is a request. For each $1 \le j \le k$, a server s_j is characterized by the position $p(s_j) \in X$ and s_j has capacity $c_j \in \mathbb{N}$, i.e., s_j can be matched with at most c_j requests. We assume that $n \le c_1 + \cdots + c_k$, so that every request can be matched with some server. For each $1 \le i \le n$, a request r_i is also characterized by the position $p(r_i) \in X$.

The set S is given to an online algorithm in advance, while requests are given one by one from r_1 to r_n. At any time of the execution of an algorithm, a server is called *free* if the number of requests matched with it is less than its capacity, and *full* otherwise. When a request r_i is revealed, an online algorithm must match r_i with one of the free servers. If r_i is matched with s_j, the pair (r_i, s_j) is added to the current matching and the cost $\text{COST}(r_i, s_j) = d(p(r_i), p(s_j))$ is incurred for this pair. The cost of the matching is the sum of the costs of all the pairs contained in it. The goal of online algorithms is to minimize the cost of the final matching. We refer to such a problem as the *online facility assignment* problem with k servers and denote it by $\text{OFA}(k, \{c_j\}_{j=1}^k)$. For the special case that $c_1 = \cdots = c_k = \ell \ge 1$, we simply denote the problem by $\text{OFA}(k, \ell)$, and call this setting *uniform capacity*.

By setting $X = \mathbb{R}$, we can regard the underlying metric as a line, and we denote such a problem by $\text{OFAL}(k, \{c_j\}_{j=1}^k)$ for general capacities and $\text{OFAL}(k, \ell)$ for uniform capacities. Note that in this case, $p(s_j)$ and $p(r_i)$ are real numbers. Without loss of generality, we assume that $p(s_1) < \cdots < p(s_k)$ and let $d_j = p(s_{j+1}) - p(s_j)$ for each $1 \le j \le k - 1$. When $d_1 = \cdots = d_{k-1} = d$ with some constant $d > 0$, we attach the subscript "eq" to the problem name to indicate that k servers are of equidistant, such as $\text{OFAL}_{eq}(k, \{c_j\}_{j=1}^k)$ and $\text{OFAL}_{eq}(k, \ell)$. For the subsequent discussion, we assume without loss of generality that $d = 1$ for both $\text{OFAL}_{eq}(k, \{c_j\}_{j=1}^k)$ and $\text{OFAL}_{eq}(k, \ell)$.

In the rest of the section, we will abuse the notations $r_i \in \mathbb{R}$ and $s_j \in \mathbb{R}$ for $\text{OFAL}(k, \{c_j\}_{j=1}^k)$ instead of $p(r_i) \in \mathbb{R}$ and $p(s_j) \in \mathbb{R}$, respectively, when it causes no confusion.

10.2.2.2 Competitive Ratio

To evaluate the performance of an online algorithm ALG, we use the (strict) competitive ratio. For $\text{OFA}(k, \{c_j\}_{j=1}^k)$, let $S = \{s_1, \ldots, s_k\}$ be the set of k servers. For an online algorithm ALG for $\text{OFA}(k, \{c_j\}_{j=1}^k)$ and a request sequence $\sigma = r_1 \cdots r_n$, we use $\text{ALG}(\sigma)$ to denote the total cost incurred by ALG when processing σ. Also, let $\text{OPT}(\sigma)$ denote the optimal cost for a request sequence σ, i.e., the total cost incurred by an optimal offline algorithm for σ. We say that ALG is *c-competitive* if $\text{ALG}(\sigma) \leq c \cdot \text{OPT}(\sigma)$ for any request sequence σ. The *competitive ratio* $\mathcal{R}(\text{ALG})$ of ALG is defined to be the infimum of $c \geq 1$ such that ALG is c-competitive, i.e., $\mathcal{R}(\text{ALG}) = \inf\{c \geq 1 : \text{ALG is } c\text{-competitive}\}$.

10.2.2.3 Technical Lemmas

As mentioned in Sect. 10.2.2.1, servers may have different capacities $c_1, c_2, \ldots, c_k$ and the condition is that $n \leq c_1 + \cdots + c_k$. However, as we show below, for the design of online algorithms for $\text{OFA}(k, \{c_j\}_{j=1}^k)$, it is sufficient to deal with the case that $n = c_1 + \cdots + c_k$ (Lemma 10.1) and that $c_1 = \cdots c_k = \ell$ (Lemma 10.2).

Lemma 10.1 *For* $\text{OFA}(k, \{c_j\}_{j=1}^k)$*, let $L = c_1 + \cdots + c_k$. Then for any $c \geq 1$ and any online algorithm* ALG*,* $\text{ALG}(\sigma) \leq c \cdot \text{OPT}(\sigma)$ *holds for any request sequence σ such that $|\sigma| = L$ iff* $\text{ALG}(\sigma') \leq c \cdot \text{OPT}(\sigma')$ *holds for any request sequence σ' such that $|\sigma'| \leq L$.*

Lemma 10.2 *For any $\ell \geq 1$, any $c_1, \ldots, c_k$ such that $1 \leq c_1, \ldots, c_k \leq \ell$, and any $c \geq 1$, there exists a c-competitive algorithm for* $\text{OFA}(k, \ell)$ *iff there exists a c-competitive algorithm for* $\text{OFA}(k, \{c_j\}_{j=1}^k)$.

Based on Lemmas 10.1 and 10.2, we can focus on the case that $c_1 = \cdots = c_k = \ell$ and the number of requests is exactly $k\ell$.

10.2.2.4 Greedy Algorithm

Finally, we give the definition of the greedy algorithm (denoted by GRDY) for $\text{OFA}(k, \ell)$.

Definition 10.1 The greedy algorithm (denoted by GRDY) matches each request with the nearest free server. If there exist at least two nearest free servers for the request r_i, then GRDY chooses the one with the *largest* index to match r_i.

10.2.3 Online Facility Assignment with a Small Number of Servers

In this subsection, we review the results in [20], which studies $\mathrm{OFA}(k,\ell)$ and $\mathrm{OFAL}_{eq}(k,\ell)$ for small k, i.e., $k = 2, 3, 4$, and 5. For $k = 2$, we have the following two theorems.

Theorem 10.1 *For any $\ell \in \mathbb{N}$, the competitive ratio of* GRDY *is at most 3 for* $\mathrm{OFA}(2,\ell)$.

Theorem 10.2 *For any $\ell \in \mathbb{N}$, the competitive ratio of any deterministic online algorithm for* $\mathrm{OFA}(2,\ell)$ *is at least 3.*

These two theorems imply that GRDY is one of the best online algorithms for $\mathrm{OFA}(2,\ell)$ in terms of the competitive ratio. Since $\mathrm{OFAL}_{eq}(2,\ell)$ is a special case of $\mathrm{OFA}(2,\ell)$, Theorem 10.1 applies also for $\mathrm{OFAL}_{eq}(2,\ell)$. Since the adversarial requests in the proof of Theorem 10.2 are constructed on line metric, we have the following corollary (note that the problem with two servers is trivially an equidistant case):

Corollary 10.1 *For any $\ell \in \mathbb{N}$, the competitive ratio of* GRDY *is at most 3 and the competitive ratio of any deterministic online algorithm is at least 3 for* $\mathrm{OFAL}_{eq}(2,\ell)$.

For $k = 3, 4$, and 5, we obtained lower bounds on the competitive ratio for $\mathrm{OFAL}_{eq}(k,\ell)$. In order to analyze the lower bound, the following notions and lemma play key roles.

Definition 10.2 (*Surrounding servers* [3, 30]) Given a request r for $\mathrm{OFAL}(k,\ell)$, the surrounding servers for r are s^L and s^R, where s^L is the closest free server to the left of r (if any) and s^R is the closest free server to the right of r (if any). If there is a free server $s \in S$ such that $s = r$, then the surrounding server of r is only s.

Definition 10.3 (*Surrounding-oriented* [3, 30]) Let ALG be an online algorithm for $\mathrm{OFAL}(k,\ell)$. We say that ALG is surrounding-oriented for a request sequence σ if it matches every request r of σ with one of the surrounding servers of r. We say that ALG is surrounding-oriented if it is surrounding-oriented for any request sequence.

Lemma 10.3 ([3, 20]) *Let* ALG *be an online algorithm for* $\mathrm{OFAL}(k,\ell)$. *Then there exists a surrounding-oriented algorithm* ALG' *for* $\mathrm{OFAL}(k,\ell)$ *such that* $\mathrm{ALG}'(\sigma) \le \mathrm{ALG}(\sigma)$ *for any σ.*

For the purpose of discussing lower bounds on the competitive ratio, it suffices by Lemma 10.3 to consider only surrounding-oriented algorithms. Using this property, we obtained the following three theorems.

Theorem 10.3 *The competitive ratio of any deterministic online algorithm for* $\mathrm{OFAL}_{eq}(3,\ell)$ *is at least* $1 + \sqrt{6}$ (> 3.449).

Theorem 10.4 *The competitive ratio of any deterministic online algorithm for* $\text{OFAL}_{eq}(4, \ell)$ *is at least* $\frac{4+\sqrt{73}}{3}$ (> 4.181).

Theorem 10.5 *The competitive ratio of any deterministic online algorithm for* $\text{OFAL}_{eq}(5, \ell)$ *is at least* $\frac{13}{3}$ (> 4.333).

10.2.4 Capacity-Insensitive Algorithms

This subsection introduces the results in [17], which studies $\text{OFA}(k, \ell)$ and $\text{OFAL}_{eq}(k, \ell)$ for general k. The competitive analysis for $\text{OFA}(k, \ell)$ for general capacity ℓ looks harder than that for $\text{OFA}(k, 1)$. In order to make the algorithm design much easier, we define the *capacity-insensitive* property [17]. We say that an online algorithm ALG has *capacity-insensitive* property if the c-competitiveness of ALG for $\text{OFA}(k, 1)$ is equivalent to the c-competitiveness of ALG for $\text{OFA}(k, \ell)$ with general ℓ.

10.2.4.1 MPFS Algorithms

To begin with, we introduce the class of MPFS (Most Preferred Free Servers) algorithms and show that any MPFS algorithm has the capacity-insensitive property (Theorem 10.6).

Definition 10.4 Let ALG be an online algorithm for $\text{OFA}(k, \ell)$. We say that ALG is an MPFS (Most Preferred Free Servers) algorithm if for any request sequence $\sigma = r_1 \cdots r_n$ such that $n = k\ell$, it behaves as follows: For each $1 \le i \le n$,

1. the priority (a total order) of servers for r_i is determined by only the position of r_i;
2. ALG matches r_i with the server with the highest priority among free servers.

Let $\mathcal{MPFS}$ be the class of MPFS algorithms. If ALG's priority is sorted by the distance from a request, then ALG is the greedy algorithm. Therefore, the class $\mathcal{MPFS}$ contains the greedy algorithm.

Theorem 10.6 *Let* ALG $\in \mathcal{MPFS}$. *For any* $c \ge 1$ *and any* $\ell \ge 1$, ALG *is* c*-competitive for* $\text{OFA}(k, 1)$ *if and only if* ALG *is* c*-competitive for* $\text{OFA}(k, \ell)$.

This theorem enables us to focus on $\text{OFA}(k, 1)$ once an online algorithm for $\text{OFA}(k, \ell)$ is confirmed to be in $\mathcal{MPFS}$.

10.2.4.2 Competitive Analysis for GRDY

Based on the above idea, we succeeded in showing the exact competitive ratio of GRDY for $\text{OFAL}_{eq}(k, \ell)$.

Theorem 10.7 *For* $\text{OFAL}_{eq}(k, \ell)$ *such that* $k \geq 2$*, the competitive ratio of* GRDY *is exactly* $4k - 5$.

Note that Theorem 10.7 generalizes Corollary 10.1 to $k > 2$. Here we show only the proof for the lower bound.

Proof (of the lower bound) For simplicity, assume that $s_j = j - 1$ for each $1 \leq j \leq k$. We construct a request sequence σ such that $\text{GRDY}(\sigma) = (4k - 5) \cdot \text{OPT}(\sigma)$. First, for each j, we give $\ell - 1$ requests to the position $j - 1$ (where the server s_j is placed), and without loss of generality, both OPT and GRDY match them with r_j with zero cost. We then give k requests $r_1 \cdots r_k$ where $r_1 = \frac{1}{2}$ and $r_j = s_j = j - 1$ for each $2 \leq j \leq k$. By Definition 10.1,

$$\begin{aligned} \text{GRDY}(\sigma) &= \frac{1}{2} + (k-2) + (k-1) = \frac{4k-5}{2}; \\ \text{OPT}(\sigma) &= \frac{1}{2}. \end{aligned}$$

Thus for the request sequence σ defined above, it follows that

$$\frac{\text{GRDY}(\sigma)}{\text{OPT}(\sigma)} = \frac{\frac{4k-5}{2}}{\frac{1}{2}} = 4k - 5$$

and this implies that $\mathcal{R}(\text{GRDY}) \geq 4k - 5$. □

10.2.4.3 Competitive Analysis for IDAS and MPFS Algorithms

Now we give an MPFS algorithm IDAS (Interior Division for Adjacent Servers) that performs better than GRDY. We show that its competitive ratio is at most $2k - 1$ for $\text{OFAL}_{eq}(k, \ell)$. We also show that IDAS is best among MPFS algorithms, by showing a lower bound on the competitive ratio of any MPFS algorithm. We first formally state the lower bound.

Theorem 10.8 *The competitive ratio of any MPFS algorithm for* $\text{OFAL}_{eq}(k, \ell)$ *is at least* $2k - 1$.

Next, we show the definition of IDAS. Before doing so, we provide several notations. Fix $a, b \in \mathbb{R}$ with $a < b$ arbitrarily. For any $x, y \in \mathbb{R}$ such that $a \leq x < y \leq b$, let $B(x, y)$ be the point that internally divides the line segment $[x, y]$ into $b - x$ to $y - a$, i.e.,

$$B(x, y) = \frac{(b-x)y + (y-a)x}{(b-x) + (y-a)} = \frac{by - ax}{b - a + y - x}.$$

Note that $x < B(x, y) < y$ and $B(x, y)$ implies a *boundary* between x and y.

Given the set $S = \{s_1, \ldots, s_k\}$ of k servers with $s_1 < \cdots < s_k$, we fix parameters $a, b \in \mathbb{R}$ such that $a \leq s_1 < s_k \leq b$. Then the algorithm $\text{IDAS}_{[a,b]}$ can be described in Algorithm 10.1.

Algorithm 10.1 $\text{IDAS}_{[a,b]}$ (Interior Division for Adjacent Servers)

```
For a request $r$, let $\text{SS}(r)$ be the set of surrounding servers for $r$.
1. If $|\text{SS}(r)| = 1$, then match $r$ with the unique surrounding server.
2. If $|\text{SS}(r)| = 2$, then let $s^L$ be the left surrounding server for $r$ and $s^R$ be the
   right surrounding server for $r$.
   (a) If $r \le B(s^L, s^R)$, then match $r$ with $s^L$.
   (b) If $B(s^L, s^R) < r$, then match $r$ with $s^R$.
```

To observe that $\text{IDAS}_{[a,b]} \in \mathcal{MPFS}$, the following property of $B(*, *)$ is useful and the boundary $B(*, *)$ naturally induces a total order $\preceq_\rho$.

Property 10.2 *Fix $a, b \in \mathbb{R}$ with $a < b$ arbitrarily. For any $x, y, z \in \mathbb{R}$ such that $a \le x < y < z \le b$, $B(x, y) < B(x, z) < B(y, z)$.*

Proof This follows from the straightforward calculations:

$$B(x,z) - B(x,y) = \frac{(b-a)(z-y)(b-x)}{(b-a+z-x)(b-a+y-x)} > 0;$$
$$B(y,z) - B(x,z) = \frac{(b-a)(y-x)(z-a)}{(b-z+y-a)(b-a+z-x)} > 0,$$

where the inequalities follow from the assumption that $a \le x < y < z \le b$. □

We define the following binary relation $\preceq_\rho$ on $[a, b]$ with a parameter $\rho \in \mathbb{R}$.

Definition 10.5 For any $a, b \in \mathbb{R}$, let $[a, b]$ be the closed interval and fix $\rho \in \mathbb{R}$ arbitrarily. For any $x, y \in [a, b]$, we write $x \preceq_\rho y$ if one of the following conditions holds: (1) $x = y$; (2) $x < y$ and $B(x, y) < \rho$; (3) $y < x$ and $\rho \le B(y, x)$.

For the binary relation $\preceq_\rho$ on $[a, b]$, the following result holds.

Theorem 10.9 *For any $\rho \in \mathbb{R}$, $\preceq_\rho$ is a total order on the closed interval $[a, b]$.*

We summarize the properties of the total order $\preceq_\rho$ in the following remark.

Remark 10.1 For any $a, b \in \mathbb{R}$ such that $a < b$, let $[a, b]$ be the closed interval. Then for any $x, y \in [a, b]$ and any $\rho \in \mathbb{R}$, the following properties hold:

(1) if $\rho < x < y$, then $y \preceq_\rho x \preceq_\rho \rho$;
(2) if $x < y < \rho$, then $x \preceq_\rho y \preceq_\rho \rho$;
(3) if $\rho < a$, then $x \preceq_\rho y$ iff $x \ge y$;
(4) if $b < \rho$, then $x \preceq_\rho y$ iff $x \le y$;
(5) if $\rho \in [a, b]$, then $x \preceq_\rho \rho$.

The property (5) implies that ρ is the maximum in $[a, b]$ w.r.t. the total order $\preceq_\rho$. □

From Remark 10.1, the following Definition 10.6 of $\text{IDAS}_{[a,b]}$ is equivalent to Algorithm 10.1. For a request r, we say that $s \in S$ is the highest free server w.r.t. $\preceq_r$ if s is free and $s' \preceq_r s$ for all free servers $s' \in S$ just before matching r to a server.

Definition 10.6 For a request sequence $\sigma = r_1 \cdots r_i \cdots r_{k\ell}$, the algorithm $\textsc{idas}_{[a,b]}$ for OFAL(k, ℓ) works as follows: For each $1 \leq i \leq k\ell$, it matches a request r_i with the highest free server $s \in S$ w.r.t. the total order $\preceq_{r_i}$.

From Definition 10.6, it is immediate that $\textsc{idas}_{[a,b]} \in \mathcal{MPFS}$. The following theorem claims that $\textsc{idas}_{[s_1,s_k]}$ is one of the best MPFS algorithms.

Theorem 10.10 *The competitive ratio of* $\textsc{idas}_{[s_1,s_k]}$ *for* OFAL$_{eq}(k, \ell)$ *is at most* $2k - 1$.

10.2.5 Conclusions

In this section, we gave overview on the competitive analysis for the online facility assignment problem on a line, which appear in [17, 20]. One of the most interesting future works is to design an online algorithm better than IDAS. Since no algorithm in $\mathcal{MPFS}$ can be better than IDAS, we need to invent a capacity-insensitive algorithms not in $\mathcal{MPFS}$.

10.3 Two-Sided Matching with Flexible Quotas: Student-Project-Resource Matching-Allocation Problem

This section presents a new framework called a student-project-resource matching-allocation problem, where students have preferences over projects and the projects have preferences over students [32]. Liu et al. [32] show that no strategyproof mechanism satisfies fairness and weak efficiency requirements, and develop a new class of strategyproof mechanisms called Sample and Deferred Acceptance (SDA), which satisfies several properties of fairness and efficiency.

10.3.1 Introduction

This section introduces a simple, but fundamental problem called student-project-resource matching-allocation problem (SPR) presented in [32]. On one hand, SPR can be considered as a two-sided, many-to-one matching problem [41] since students are matched to projects based on their preferences. On the other hand, it is also a discrete resource allocation problem [29] since resources are allocated to each project. However, unlike the standard setting of two-sided many-to-one matching, where the capacity of each project is exogenously determined, we assume the capacities are endogenously determined by resource allocation.

If the mechanism designer knows the preferences of the students, she can allocate resources to projects using combinatorial optimization techniques. If each project's capacity is determined, even if the mechanism designer does not know the students' preferences beforehand, she can find a matching that satisfies desirable properties with a strategyproof mechanism, e.g., Deferred Acceptance mechanism (DA) [11], such that students voluntarily disclose their true preferences. However, the mechanism designer usually does not know their preferences. Thus, a common practice for solving this problem is to determine its resource allocation part based on the expectations or the past data and set the capacities of the projects. Then the actual matching of students to projects is determined by a matching mechanism. In this approach, if the expectations used in the first problem are incorrect, the outcome can be sub-optimal; excess demand and supply for seats can coexist, which can be avoided by better resource allocation.

One instance where such a practice is applied is a school choice program for assigning students to public schools. In a standard setting, each school has a maximum quota that is determined in advance. Assume a local government (e.g., a city/prefecture/state) has spare resources, e.g., sufficient budget to hire temporary teachers, which can be allocated based on the demand. Then the maximum quota of each school is no longer fixed in advance, but it can be flexibly modified based on the actual demand utilizing extra budget/resources.

This article follows previous works that address constrained matching problems. Two-sided matching has attracted considerable attention from AI and theoretical computer science researchers. A standard market deals with maximum quotas, i.e., capacity limits that cannot be exceeded. However, many real-world matching markets are subject to a variety of distributional constraints [28], including regional maximum quotas, which restrict the total number of students assigned to a set of schools [24], minimum quotas, which guarantee that a certain number of students are assigned to each school [9, 14, 16], and diversity constraints [7, 15, 27, 31, 46]. The rest of this section is organized as follows. Section 10.3.2 introduces a model of an SPR. Section 10.3.3 shows that no mechanism is fair, weakly nonwasteful, resource efficient, and strategyproof. Section 10.3.4 introduces a class of mechanisms called Sample and Deferred Acceptance (SDA). Finally, Sect. 10.3.5 concludes the section.

10.3.2 Model

A student-project-resource matching-allocation problem (SPR) is defined as follows:

Definition 10.7 (*Student-Project-Resource allocation (SPR) Instance*) An SPR instance is a tuple $(S, P, R, \succ_S, \succ_P, q_R, T_R)$.

- $S = \{s_1, \ldots, s_{|S|}\}$ is a set of students.
- $P = \{p_1, \ldots, p_{|P|}\}$ is a set of projects.
- $R = \{r_1, \ldots, r_{|R|}\}$ is a set of indivisible resources.

- $\succ_S = (\succ_s)_{s \in S}$ are the student's strict preferences over set $P \cup \{\emptyset\}$. Symbol $\emptyset$ means that a student is not assigned to any project.
- $\succ_P = (\succ_p)_{p \in P}$ are the projects' strict preferences over set $S \cup \{\emptyset\}$. Symbol $\emptyset$ means that a project is assigned no student.
- $q_R = (q_r)_{r \in R}$ are the capacities of resources; $q_r \in \mathbb{N}_{>0}$ for every $r \in R$.
- $T_R = (T_r)_{r \in R}$ is a profile of the resource compatibility lists, where each $T_r \subseteq P$ is a set of projects to which resource r can be allocated. Since resource r is indivisible, it must be allocated to exactly one project in T_r.

Note that since we assume a resource indivisible, we cannot allocate a resource r with the capacity of two to the projects p_1 and p_2 with the capacity of one each. We illustrate our setting with the following example.

Example 10.1

There are four students, s_1, s_2, s_3, s_4, four projects, p_1, p_2, p_3, p_4, and two resources, r_1, r_2, where $T_{r_1} = \{p_1, p_2\}$, $T_{r_2} = \{p_3, p_4\}$, and $q_{r_1} = 2$, $q_{r_2} = 1$. The following are their preferences:

$$\begin{aligned}
&s_1 : p_1 \succ p_2 \succ p_4 \succ p_3 \succ \emptyset, \; p_1 : s_4 \succ s_3 \succ s_2 \succ s_1 \succ \emptyset,\\
&s_2 : p_1 \succ p_2 \succ p_3 \succ p_4 \succ \emptyset, \; p_2 : s_4 \succ s_3 \succ s_2 \succ s_1 \succ \emptyset,\\
&s_3 : p_1 \succ p_2 \succ p_3 \succ p_4 \succ \emptyset, \; p_3 : s_1 \succ s_2 \succ s_3 \succ s_4 \succ \emptyset,\\
&s_4 : p_4 \succ p_3 \succ p_2 \succ \emptyset \succ p_1, \; p_4 : s_1 \succ s_2 \succ s_3 \succ s_4 \succ \emptyset.
\end{aligned}$$

Since we assume resources are indivisible, it is impossible to allocate students to three different projects although the total capacity of all the resources equals three. A resource can only be allocated to a compatible project, e.g., r_1 can be allocated to either p_1 or p_2. The following are the possible capacities of the four projects: $(2, 0, 1, 0)$, $(2, 0, 0, 1)$, $(0, 2, 1, 0)$, or $(0, 2, 0, 1)$.

We follow the matching with a contracts model [18]. Contract $(s, p) \in S \times P$ means that student s is matched to project p. Contract (s, p) is acceptable to student s (resp. project p) if $p \succ_s \emptyset$ holds (resp. $s \succ_p \emptyset$). Let X denote the set of all contracts that are acceptable to the projects.[2]

A matching is a set of contracts satisfying the following conditions.

Definition 10.8 (*Matching*) A matching is a subset $Y \subseteq X$, where for every student $s \in S$, $Y_s = \{(s, p) \mid (s, p) \in X\}$, either $|Y_s| = 0$ or $Y_s = \{(s, p)\}$ and $p \succ_s \emptyset$ hold.

For matching Y, let $Y(s)$ denote the project to which s is matched ($Y(s) = \emptyset$ if s is not matched to any project in Y), and let $Y(p) \subseteq S$ denote the set of students assigned to project p ($Y(p) = \emptyset$ means no student is allocated to p in Y).

[2] In designing a strategyproof mechanism, we assume student preferences are private information, and the other information is public. Thus, we assume X, i.e., the set of all contracts that are acceptable to the projects is public.

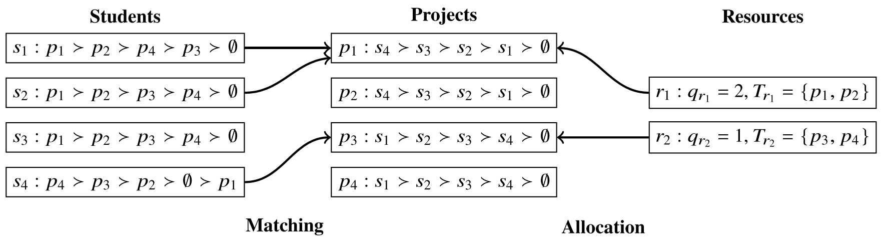

Students
Projects
Resources
$s_1 : p_1 \succ p_2 \succ p_4 \succ p_3 \succ \emptyset$
$s_2 : p_1 \succ p_2 \succ p_3 \succ p_4 \succ \emptyset$
$s_3 : p_1 \succ p_2 \succ p_3 \succ p_4 \succ \emptyset$
$s_4 : p_4 \succ p_3 \succ p_2 \succ \emptyset \succ p_1$
$p_1 : s_4 \succ s_3 \succ s_2 \succ s_1 \succ \emptyset$
$p_2 : s_4 \succ s_3 \succ s_2 \succ s_1 \succ \emptyset$
$p_3 : s_1 \succ s_2 \succ s_3 \succ s_4 \succ \emptyset$
$p_4 : s_1 \succ s_2 \succ s_3 \succ s_4 \succ \emptyset$
$r_1 : q_{r_1} = 2, T_{r_1} = \{p_1, p_2\}$
$r_2 : q_{r_2} = 1, T_{r_2} = \{p_3, p_4\}$
Matching
Allocation

Fig. 10.1 SPR instance: matching $\hat{Y}$ and allocation $\hat{\mu}$ in Example 10.1

In an SPR, we also need to describe how resources are allocated to projects. A matching's *feasibility* is defined based on this description.

Definition 10.9 (*Allocation*) An allocation $\mu : R \to P$ maps each resource r to a project $\mu(r) \in T_r$. Let $q_\mu(p) = \sum_{r \in \mu^{-1}(p)} q_r$.[3]

In other words, a project's maximum quota is defined by the sum of all resources that are allocated to it. Note that multiple resources can be allocated to a single project.

Definition 10.10 (*Feasibility*) A feasible matching (Y, μ) is a matching-allocation pair where $|Y(p)| \le q_\mu(p)$ holds for every $p \in P$.

In the setting of Example 10.1, assume matching $\hat{Y}$ is $\{(s_1, p_1), (s_2, p_1), (s_3, \emptyset), (s_4, p_3)\}$ and allocation $\hat{\mu}$ distributes r_1 to p_1 and r_2 to p_3. Then $(\hat{Y}, \hat{\mu})$ is a feasible matching. See Fig. 10.1 for an illustration.

Next, we introduce a concept related to efficiency called *nonwastefulness*. First, we define a situation where a student claims that the current matching is inefficient since her welfare can be improved without disadvantaging other students.

Definition 10.11 (*Claiming an Empty Seat with* μ) Given feasible matching (Y, μ), student s claims an empty seat in project p with μ if the following conditions hold:

- $p \succ_s Y(s)$ and
- $Y \setminus \{(s, Y(s))\} \cup \{(s, p)\}$ is feasible with μ.

In other words, student s claims an empty seat in project p with μ if she can move to p from current project $Y(s)$ (which can be $\emptyset$) in current allocation μ.

Definition 10.12 (*Nonwastefulness*) Given feasible matching (Y, μ), student s *possibly* claims an empty seat in project p if $\exists \mu'$ such that s claims an empty seat in p with μ'. A feasible matching (Y, μ) is nonwasteful if no student possibly claims an empty seat in any project.

[3] For $\mu^{-1}(p) = \emptyset$, we assume $q_\mu(p) = 0$.

In other words, s possibly claims an empty seat in p if s can be moved to a more preferred project p without changing the assignment of the other students with allocation μ'. Note that μ' can be different from μ. Thus, s can possibly claim an empty seat in p even if it is impossible to move her to p with current allocation μ, as long as it becomes possible with a different/better allocation μ'. In a traditional setting, since each project's maximum quota (capacity limit) is fixed, it suffices to check whether a student can be moved to another project under the fixed maximum quotas. In contrast, in our setting, maximum quotas are endogenous and flexible. Thus, the definition of nonwastefulness is modified to reflect this flexibility. In the setting of Example 10.1 in Fig. 10.1, s_4 cannot claim an empty seat in p_4 in current allocation $\hat{\mu}$ because no resource is allocated to p_4 and it is impossible to move her from p_3 to p_4. However, she possibly claims an empty seat in p_4 since by allocating r_2 to p_4, we can move her to p_4 without disadvantaging other students. Thus, $(\hat{Y}, \hat{\mu})$ does not satisfy nonwastefulness.

Next, we introduce a concept called *fairness*.

Definition 10.13 (*Fairness*) Given feasible matching (Y, μ), student s has justified envy toward student s' if for project p such that $s' \in Y(p)$, $p \succ_s Y(s)$, and $s \succ_p s'$ hold. A feasible matching (Y, μ) is fair if no student has justified envy.

In other words, student s has justified envy toward s' if s' is assigned to project p although s prefers p over her current project $Y(s)$ and project p prefers s over s'.

Next, we formally define a mechanism and introduce the desirable properties a mechanism should satisfy: strategyproofness.

Definition 10.14 (*Mechanism*) Given any SPR instance, a mechanism outputs a feasible matching (Y, μ). If a mechanism always yields a feasible matching that satisfies property A (e.g., fairness), we say that this mechanism is A (e.g., fair).

Definition 10.15 (*Strategyproofness*) A mechanism is strategyproof if no student has an incentive to misreport her preference.

10.3.3 Impossibility Theorems

In an SPR, resources should be flexibly allocated to projects for more efficient matching. However, such flexibility is hard to combine with fairness. First, we show that fairness and nonwastefulness are incompatible.

Theorem 10.11 *An SPR instance exists where no feasible matching is fair and nonwasteful.*

Proof Consider the following SPR instance[4]: two students, s_1, s_2, two projects, p_1, p_2, and a unitary resource compatible with both. The student preferences are

[4] This example is identical to the example [24], which shows that a strongly stable matching may not exist with regional quotas.

$p_1 \succ_{s_1} p_2 \succ_{s_1} \emptyset$ and $p_2 \succ_{s_2} p_1 \succ_{s_2} \emptyset$. The project preferences are $s_2 \succ_{p_1} s_1 \succ_{p_1} \emptyset$ and $s_1 \succ_{p_2} s_2 \succ_{p_2} \emptyset$. By symmetry, we can assume the resource is allocated to p_1 w.l.o.g. For fairness, s_2 must be allocated to p_1. Then s_1 possibly claims an empty seat in p_2 since moving her to p_2 is possible by allocating the resource to p_2. □

Given this impossibility theorem, let us introduce weaker conditions on efficiency.

Definition 10.16 (*Weak Nonwastefulness*) For feasible matching (Y, μ), student s strongly claims an empty seat if $Y(s) = \emptyset$ and s claims an empty seat in project p with μ. A feasible matching is weakly nonwasteful if no student strongly claims an empty seat.

In the setting of Example 10.1, feasible matching $(\hat{Y}, \hat{\mu})$ in Fig. 10.1 is weakly nonwasteful because s_3 cannot strongly claim an empty seat. Although $\hat{Y}(s_3) = \emptyset$, she cannot be assigned to any project with current allocation $\hat{\mu}$.

Definition 10.17 (*Very Weak Nonwastefulness*) For feasible matching (Y, μ), student s *very strongly claims an empty seat* if $Y(s) = \emptyset$, and $\forall \mu'$, such that (Y, μ') is feasible, $\exists p$ in which s claims an empty seat with μ'. A feasible matching is very weakly nonwasteful if no student very strongly claims an empty seat.

In other words, student s very strongly claims an empty seat if she is currently unassigned, and under *any* feasible resource allocation μ', project p exists such that s claims an empty seat in p with μ'. Note that p can be different for each μ'.

If student s very strongly claims an empty seat, she is currently unassigned and claims an empty seat in project p with current allocation μ, and thus she also strongly claims an empty seat. If she claims an empty seat in p under the current assignment, she also possibly claims an empty seat in p. Thus, nonwastefulness implies weak nonwastefulness, and weak nonwastefulness implies very weak nonwastefulness.

To define another concept called *resource efficiency*, let us first define *unanimous preferences*.

Definition 10.18 (*Unanimous Preference*) Students unanimously prefer p over p' if for every $s \in S$, $(s, p) \in X$, and $p \succ_s p'$ hold.

This condition means that project p accepts all students and all students prefer p over p'. If students unanimously prefer p over p', allocating any resource (which is compatible with both p and p') to p' is *inefficient* in terms of students' welfare. The following formalizes this intuition.

Definition 10.19 (*Resource Efficiency*) Resource allocation μ is resource efficient if no resource r, such that $p, p' \in T_r$ and students unanimously prefer p over p', is allocated to p'. A mechanism is resource efficient if it always returns a resource efficient allocation.

Now we are ready to introduce another impossibility theorem.

Theorem 10.12 *No mechanism exists that is fair, very weakly nonwasteful, resource efficient, and strategyproof.*

Proof Consider the following situation: three students, s_1, s_2, s_3, three projects, p_1, p_2, p_3, one resource, r with $q_r = 2$, and $T_r = \{p_1, p_2, p_3\}$. The following are their preferences:

$$\begin{aligned} s_1 &: p_2 \succ p_3 \succ p_1 \succ \emptyset, & p_1 &: s_1 \succ s_2 \succ s_3 \succ \emptyset, \\ s_2 &: p_3 \succ p_1 \succ p_2 \succ \emptyset, & p_2 &: s_2 \succ s_3 \succ s_1 \succ \emptyset, \\ s_3 &: p_1 \succ p_2 \succ p_3 \succ \emptyset, & p_3 &: s_3 \succ s_1 \succ s_2 \succ \emptyset. \end{aligned}$$

Recall that since all the resources must be distributed, resource r must be allocated to a project. From very weak nonwastefulness and fairness, the following are the possible matchings: allocating s_1 and s_2 to p_1, allocating s_2 and s_3 to p_2, or allocating s_3 and s_1 to p_3. From the symmetry, we can assume r is allocated to p_1 and s_1 and s_2 are assigned to p_1 w.l.o.g. Next, we examine the case where the preference of s_3 is changed to $p_3 \succ p_1 \succ p_2 \succ \emptyset$. From resource efficiency, r cannot be allocated to p_1 since all students prefer p_3 over p_1. If r is allocated to p_2 (or p_3), then from fairness and very weak nonwastefulness, s_3 must be assigned to p_2 (or p_3). This violates strategyproofness since s_3 is not assigned to any project in the original situation. □

10.3.4 Sample and Deferred Acceptance (SDA) Mechanism

Theorem 10.12 shows that fairness cannot be achieved without significantly sacrificing efficiency. To establish a good balance between fairness and efficiency in SPR problems, Liu et al. [32] introduce a new class of mechanisms called *Sample and Deferred Acceptance* (SDA). Before describing SDA, let us introduce two mechanisms, which are utilized in SDA. First, the Serial Dictatorship mechanism (SD) uses a serial order among the students. Although the order can be arbitrary, it must be determined independently of student preferences to guarantee strategyproofness. W.l.o.g., we assume this order is $s_1, s_2, \ldots$. In SD, a student is chosen based on the serial order. Each student is assigned to her most preferred project p, such that a feasible allocation exists for the (partial) matching obtained so far. This mechanism is clearly strategyproof and nonwasteful, but it is very unfair since it does not use projects' preferences at all. Next, we briefly describe the well-known Deferred Acceptance (DA) [11]. In DA, each student first applies to her most preferred project. Then each project provisionally accepts students up to its capacity limit based on its preference and rejects the rest of them. A rejected student applies to her second choice. Each project provisionally accepts students who have applied without distinguishing between newly applied and already provisionally accepted students, and so forth. To apply DA, the maximum quota (i.e., capacity limit) of each project must be predetermined. As long as maximum quotas are determined independently from students' preferences, it is strategyproof and nonwasteful.

Now, we are ready to introduce SDA.

Mechanism 10.1 (Sample and Deferred Acceptance (SDA))

Step 1: Select $S' \subseteq S$ independently of $\succ_S$, which we call sampled students, and $S' \neq \emptyset$. We call $S \setminus S'$ the regular students. Then run SD and find (partial) matching $Y_{S'}$ for S'.

Step 2: Allocate $R' \subseteq R$ to projects such that $Y_{S'}$ is feasible and R' is minimal: no $R'' \subsetneq R'$ makes $Y_{S'}$ feasible. Then allocate $R \setminus R'$ based on the preferences of S'.

Step 3: Run DA for $S \setminus S'$. The capacity of p is $q_\mu(p) - |Y_{S'}(p)|$, where μ is the resource allocation determined in Step 2.

The entire mechanism is carefully designed to guarantee strategyproofness. We adopt a popular technique used in auction domains to ensure strategyproofness, where a mechanism has some parameters and their selection affects participants' welfare [4, 13]. The idea is first dividing students/participants into two groups, then extracting the information from one group, and lastly setting the parameters of the mechanism applied to the other group based on the obtained information.

By using different methods to determine resources in Step 2, we can create a variety of sample-based mechanisms that can strike a good balance between fairness and efficiency by slightly sacrificing fairness to improve efficiency. We introduce a first strategyproof sample-based mechanism, which we call Sample and Deferred Acceptance with Voting (SDA-V). In Step 2 of Mechanism 10.1, SDA-V determines resource allocation μ based on the voting among sampled students. More specifically, each sampled student (hypothetically) votes for all projects based on $\succ_s$, where each project obtains a Borda score, i.e., the top project obtains $|P|$, the second project obtains $|P| - 1$, and so on. Then for each project, the sum of these scores is calculated. Finally, each resource r is allocated to the project that obtains the highest total score within T_r. The details of this voting procedure do not affect SDA-V's theoretical properties, e.g., whether a student can vote for a project to which she is unacceptable, or how ties are broken. Thus, they can be arbitrarily determined.

SDA-V uses a simple voting scheme for deciding resource allocation. However, if students are divided into several groups in terms of their preferences, we might obtain a sub-optimal result. Actually, the Borda count is somewhat too coarse-grained to determine an appropriate resource allocation. More specifically, for each resource r, we choose a single winner among projects T_r in SDA-V. By using the Borda count, we can choose a broadly acceptable project among candidates T_r.

Liu et al. [32] develop yet another way for deciding resource allocation, which does not rely on any voting scheme. Instead of aggregating the preference by voting, we create copies of sampled students and run a simulation for them to decide the resource allocation. More specifically, we (imaginarily) perform an SD mechanism for these copied students and obtain a resource allocation. We call this mechanism Sample and Deferred Acceptance with Simulation (SDA-S). Compared to SDA-V, SDA-S fully utilizes the preference of each sampled student, i.e., not only her first-choice project, but also her second-choice, third-choice, and so on.

In SDA-S, the detailed procedure of Step 2 in Mechanism 10.1 is given as follows:

1. Make $|S| - |S'|$ copies of sampled students (i.e., for each sampled student, we create either $\lfloor \frac{|S|-|S'|}{|S'|} \rfloor$ or $\lceil \frac{|S|-|S'|}{|S'|} \rceil$ copies).

2. Run SD for these copies to determine the resource allocation. First, we create copied students and the serial order of them is determined as follows. Assume sampled students are $s_1, s_2, \ldots, s_{|S'|}$. Let d_j^i denote the j-th copy of student s_i. Then the serial order is $d_1^1, d_1^2, \ldots, d_1^{|S'|}, d_2^1, d_2^2, \ldots, d_2^{|S'|}, d_3^1, \ldots$, i.e., using a round-robin order among the sampled students to determine the serial order of the copied students. Second, we run SD for these copied students to find an (imaginary) matching that is feasible with the remaining resources. The resources allocated by the imaginary matching are used for DA in Step 3 of Mechanism 10.1.

We describe that the family of SDA mechanisms satisfy several fundamental desiderata.

Theorem 10.13 *Any SDA instance is strategyproof, weakly nonwasteful, fair among students in $S \setminus S'$, and no sampled student has justified envy toward another regular student.*

Proof First, the sampled students S' are chosen independently of students' preferences (e.g., by random sampling). Thus, students cannot affect who will be selected in S'. Then, for each student in S', she has no incentive to misreport since her assignment is determined by SD, and the resource allocation for students $S \setminus S'$ (which is determined using her preference) is irrelevant to her. Next, for each student in $S \setminus S'$, the capacity of each project is determined independently of her preference. Also, since DA is strategyproof [6, 40], she has no incentive to misreport her preference.

Assume student s is matched to p (which can be $\emptyset$). She applied to project p', which outranks p, and was rejected. If $s \in S'$, then no feasible allocation μ' exists such that s can be assigned to p'. If $s \in S \setminus S'$, s cannot be assigned to p' with current allocation μ. Hence, any SDA instance is weakly nonwasteful.

Regarding fairness, since DA is fair [11], no regular student has justified envy toward another regular student. If sampled student $s \in S'$ is rejected by p, then no more students can be assigned to p. Thus, s never has justified envy toward a regular student who is assigned after s. □

Whether an SDA instance satisfies resource efficiency depends on how the resources are allocated based on the preferences of the sampled students. We show that the procedures for allocating resources in SDA-V and SDA-S are carefully designed such that they satisfy resource efficiency.

Theorem 10.14 *SDA-V and SDA-S are resource efficient.*

Proof Assume students unanimously prefer p over p'. When assigning sampled students S', students apply to p before applying to p'. Thus, we do not require that any resource r is allocated to p' such that $p, p' \in T_r$ holds. This property holds when assigning copied students by SD in SDA-S. Furthermore, p' never wins in the voting procedure in SDA-V. Thus, these mechanisms satisfy resource efficiency. □

10.3.5 Conclusions

This section introduced the SPR model presented by Liu et al. [32], where the maximum quota of each project is determined endogenously via resource allocation, as well as a general class of mechanisms called SDA. Besides the theoretical results presented in this article, Liu et al. [32] perform experimental evaluations to illustrate that SDA strikes a good balance between fairness and efficiency.

10.4 Formalizing Station-Based One-Way Car Sharing as Three-Sided Matching

This section presents a matching mechanism for station-based one-way car-sharing services. This service model allows users to travel one way and pick up and drop off cars at any station in a dedicated service area.

10.4.1 Background

One-way car sharing provides high flexibility of usage to users, but on the other hand, it causes an imbalance between the distribution of available vehicles and origin-destination (OD) demand, which ends up decreasing service efficiency. Many studies have tried to solve this problem by pricing usage [8, 21, 36, 45, 48] and predicting demand [5, 42, 49].

In price-based control, price elasticity is assumed. However, it can change from time to time. If it differs from the assumption at the time of price calculation, realizing the expected level of efficient utilization of vehicles is difficult. The mechanism proposed by [43] applied an extension of the budgeted multi-armed bandit algorithm to bike sharing. It tries to find an appropriate amount of monetary incentives to users to encourage them to pick up a vehicle at stations with vehicles higher than a threshold and drop off a vehicle at stations with vehicles lower than the threshold. Their mechanism can mitigate the problem of the difficulty of obtaining accurate predictions. However, it still assumes the demands change gradually.

To overcome this difficulty, we formulate the problem as a stable matching problem, i.e., we consider car-sharing regulation as a problem in finding an appropriate user-vehicle match based on user preferences. This formalization enables us to utilize many research results on the stable matching problem, such as stability, efficiency, and fairness lessons. However, many existing studies on stable matching assume that all participants appear in advance. Thus, it needs to devise a mechanism to apply them to mobility matching. Some studies have dealt with mobility matching as a stable matching problem [47, 52], but their target is limited to ride-sharing. That is, there are still few that formulate vehicle sharing as a matching problem.

To solve the imbalance problem in car sharing, we introduce the drop-off station adjustment and formulate it as a three-sided matching problem. In conventional car-sharing systems, a drop-off user selects a drop-off station, and then a pick-up user chooses which station to rent the vehicle. Here, vehicle usage can be viewed as vehicle relocation for other pick-up users. Thus if a pick-up user indirectly asks drop-off users to place the car closer to their current location, and drop-off users may change their initially intended drop-off stations, it may increase the vehicle usage level. To realize such coordination, we simultaneously consider the drop-off users' and pick-up users' preferences and try to find a matching between a drop-off user and a pick-up user via a station. We assume that a drop-off user has a preference for stations. A station has a preference for pick-up users. A pick-up user has a preference for a combination of vehicles and stations. Under this assumption, we apply a three-sided matching mechanism to find a stable matching.

The previous mechanisms, such as [8, 21, 36, 48], can be considered as centralized regulation in that the operator examines the aggregated demands and supplies and then designs adequate incentives assuming that demand predictions are accurate. In contrast, our mechanism can also be viewed as centralized regulation but examines the individual preferences of pick-up and drop-off users. To the best of our knowledge, our mechanism is the first regulation strategy for station-based one-way car-sharing systems that realize (1) stable matching for pick-up users appearing dynamically and (2) no requirement for monetary transfer, which mitigates the problem of accurate prediction for price elasticity.

10.4.2 Problem Setting and Formulation

We formulate a three-sided matching problem for car sharing regulation, including drop-off agents, stations, and pick-up agents. Here, agents driving cars are drop-off agents. The set of drop-off agents, stations, and pick-up agents are represented by $\mathcal{B}$, $\mathcal{S}$, and $\mathcal{A}$, respectively. Drop-off agent b_j has his preference on stations, $P(b_j \Rightarrow \mathcal{S})$. For example, their types consist of the distance from the drop-off station to their final destination. Pick-up agent a_t has her preference on the tuple of stations and vehicles, $P(a_t \Rightarrow \mathcal{B} \times \mathcal{S})$. For example, their preferences include the vehicle types θ_j, such as size and radar cruise control availability. Also, station s_i has its preference on pick-up agents, $P(s_i \Rightarrow \mathcal{A})$, which the system operator sets. For example, their preferences consist of the waiting time length of pick-up agents and the scarcity of vehicles at the destination of pick-up agents.

We assume a road network $\mathcal{L}$ consisting of nodes and links and parking stations are located at some nodes. There are n_{st} stations. Station $s_i \in \mathcal{S}$ has n_{s_i} parking spots. Different parking stations may have different numbers of parking spots. Cars can be parked at any station with at least an unoccupied parking spot.

The car-sharing system operates n_v cars in a dedicated service area. The location l of a car corresponds to positions of the road networks $l \in \mathcal{L}$. We use a discrete-time model, $t = 0, 1, 2, \cdots$. At $t = 0$, the status of each vehicle is "driving" or "parked" at some locations.

At each time from $t = 1$, at most one pick-up agent a_t appears. We assume that if more than one pick-up agents demand car usage simultaneously, these requests are sequenced in an arbitrary order by reducing the time resolution. Pick-up agent a_t wants to leave location $l^o_{a_t}$ at time t' ($t' > t$) and move to $l^d_{a_t}$. Agents will show up ($t' - t$)-time units before the desired usage time. If pick-up agent a_t realizes the intended trip by using the vehicle whose type is θ_j, she obtains positive valuation $v_{a_t}(\theta_j)$. Also, pick-up agent a_t has disutility $-f_{a_t}(x)(\leq 0)$ depending on the elapsed time x from t' required to pick up a vehicle by walking from $l^o_{a_t}$ to the station location or by waiting for the arrival of the arranged car. Pick-up agent a_t may use a parked car. We assume a virtual drop-off agent b_j for each parked car. Pick-up agent a_t's preference list on the tuple of stations and vehicles can be composed of $v_{a_t}(\theta_j)$ and $-f_{a_t}(x)$. We assume that the preference list has no tie.

Agents driving cars are drop-off agents. $\mathcal{B}_t$ represents the set of drop-off agents at time t. Drop-off agent $b_j \in \mathcal{B}_t$ whose destination is $l^d_{b_j}$ tries to drop off the vehicle at the station $l^{d'}_{b_j}$ nearest to $l^d_{b_j}$ and walk to $l^d_{b_j}$. His drop-off station might be different from the nearest station to his destination because the nearest station is occupied, or he enters into a contract with a pick-up agent. In such cases, he has disutility $-g_{b_j}(x)(\leq 0)$ by the delay in arriving at the destination $l^d_{b_j}$, i.e., the difference in time x between the new and original itineraries. Because the driving speed is faster than the walking speed, if he changes his itinerary, x is larger than zero. After pick-up agent a_t starts using a vehicle, she changes its role to drop-off agent $b_{j'}$. We allow $f_{a_t}(x) \neq g_{b_{j'}}(x)$. This is because the pick-up agent may change her trip plan, e.g., she can use public transportation instead or cancel her trip before departure but cannot do so once she has started using the vehicle. Drop-off agent b_j's preference list on stations can be composed of $-g_{b_j}(x)$. We assume that the preference list has no tie. For virtual agent b_j, its preference list has one element of the currently parked station.

Station s_i's preference list on pick-up agents is given by the system operator. The system operator can learn pick-up agents' current locations and the time their usage request is reported. We assume that the preference list has no tie.

For pick-up agent a_t, (b_j, s_k, a_t) denotes the matching contract under which drop-off agent b_j drops off his car at station s_k, which might be different from the original drop-off station $l^{d'}_{b_j}$. b_j might be a virtual drop-off agent corresponding to a vehicle parked at station s_k. $t_{a_t}((b_j, s_k, a_t))$ is a function that returns the time from t' required to pick up a vehicle under matching contract (b_j, s_k, a_t) for pick-up agent a_t. $t_{b_j}((b_j, s_k, a_t))$ is a function that returns the delay in arriving at the final destination for drop-off agent b_j. Here, we assume that the agents and the operators have accurate information on the trip duration between two locations either by walking or driving.

If matching contract (b_j, s_k, a_t) is entered into, the utilities for pick-up agent a_t and drop-off agent b_j are represented as follows.

$$u_{a_t} = v_{a_t}(\theta_j) - f_{a_t}(t_{a_t}((b_j, s_k, a_t))) \tag{10.1}$$

$$u_{b_j} = -g_{b_j}(t_{b_j}((b_j, s_k, a_t))) \tag{10.2}$$

The arguments in the utility functions are measured by time. For example, if a car is dropped off at a station near the current location of a pick-up agent, but the car's arrival at the station takes a long time, the pick-up agent's utility would decrease, and she would not confirm such a contract. Once the matching contract is entered into, neither the pick-up agent nor the drop-off agent is allowed to break the contract. Our objective is to find a matching that maximizes a metric the system operator specifies.

10.4.3 Proposed Mechanism

We propose a novel regulation mechanism for station-based one-way car-sharing systems. The imbalance between vehicle distribution and demand will be reduced through the pick-up agent-station-drop-off agent contract by executing a three-sided matching mechanism, i.e., the adjustment of the drop-off/pick-up station. We assume the system operator holds information on current locations and intended drop-off locations for pick-up agents.

So far, a three-sided matching problem has been studied. In this paper, we extend the mechanism of Zhang and Zhong [51]. They discussed the case where the way of preference is restricted as follows. There are three groups $\mathcal{B}$, $\mathcal{S}$, and $\mathcal{A}$. An element belonging to $\mathcal{B}$ has a preference for each element of $\mathcal{S}$. An element belonging to $\mathcal{S}$ has a preference for each element of $\mathcal{A}$. An element belonging to $\mathcal{A}$ has a preference for a product of each element of $\mathcal{B}$ and each element of $\mathcal{S}$.

In the context of car sharing, $\mathcal{B}$, $\mathcal{S}$, and $\mathcal{A}$ correspond to drop-off agents, stations, and pick-up agents, respectively. A drop-off agent prefers a station that minimizes the time required to reach the final destination. The station has a preference for pick-up agents based on the service policy of the operator. A pick-up agent has a preference for a combination of cars and stations. Here, although $\mathcal{B}$ is a set of drop-off agents, a car's type, such as its size and equipment, can be specified once a drop-off agent is chosen. A pick-up agent can express her preference on the tuple of $\mathcal{B} \times \mathcal{S}$.

In our mechanism, we introduce a batch size. A batch size means how often the matching mechanism is executed. In the conventional setting of the matching problem, all the participants appear at the beginning. On the other hand, pick-up agents appear sequentially in our setting. Thus, the batch size is a control parameter in our mechanism. If the batch size is set to one, i.e., matching is executed whenever a pick-up agent appears, our mechanism is close to the no-regulation case. This is because few drop-off agents can participate in the matching process. The larger the batch size, the greater the room for adjustment. On the other hand, it asks pick-up agents to declare their preference list in advance, reducing trip planning flexibility.

Mechanism 10.2 (Drop-off/Pick-up Station Adjustment)

Step 1: Set batch size to n_{ba}.
Step 2: Wait until receiving n_{ba} usage requests.
Step 3: Determine the set of drop-off agents $\mathcal{B}_{driving}$ and the set of virtual drop-off agents $\mathcal{B}_{parked}$. Let $\mathcal{B} = \mathcal{B}_{driving} \cup \mathcal{B}_{parked}$.
Step 4: All $b \in \mathcal{B}$ is active, and let $M = \emptyset$. M represents the set of accepted matching.
Step 5: For each active agent $b \in \mathcal{B}_{driving}$, according to the preference list $P(b \Rightarrow \mathcal{S})$, propose the most preferable active one in $\mathcal{S}$ and mark it as s. For each active agent $b \in \mathcal{B}_{parked}$, propose the one in $\mathcal{S}$ that b is currently parked and mark it as s. Thus, we get a pair (b, s), and all pairs obtained in this way are denoted as K. Let $K(s) = \{b \in \mathcal{B} | (b, s) \in K\}$, that is, all $b \in K(s)$ proposed to s.
Step 6: For the resulting pairs $(b, s) \in K$, according to the preference list $P((b, s) \Rightarrow \mathcal{A})$, which is virtually equal to $P(s \Rightarrow \mathcal{A})$, propose the most preferable one in $\mathcal{A}$ and mark it as a. Thus, we get a triple (b, s, a), and all triples obtained in this way are denoted as Q. Note that any two different elements (b, s, a) and (b', s', a') in Q, there must be $b \neq b'$. Let $R = Q \cup M$, and $R|_{\mathcal{A}} = \{a \in \mathcal{A} | (b, s, a) \in R\}$.
Step 7: For each $\bar{a} \in R|_{\mathcal{A}}$ according to the preference list $P(\bar{a} \Rightarrow \mathcal{B} \times \mathcal{S})$, choose the most preferable one in all the pairs proposed to it (if $M_{\mathcal{B}\times\mathcal{S}}(\bar{a}) = (b'', s'')$, then (b'', s'') is included) and mark it as $(\bar{b}, \bar{s})$. Reject the other pairs (b', s') and remove $\bar{a}$ from the preference list $P((b', s') \Rightarrow \mathcal{A})$.
Step 8: If $M_{\mathcal{B}\times\mathcal{S}}(\bar{a}) = \emptyset$, then let $M := M \cup \{(\bar{b}, \bar{s}, \bar{a})\}$, and marked $\bar{b}$ and $\bar{s}$ as inactive. If $M_{\mathcal{B}\times\mathcal{S}}(\bar{a}) = (b'', s'')$ and $(\bar{b}, \bar{s}, \bar{a}) \neq (b'', s'', \bar{a})$, then let $M = M \cup \{(\bar{b}, \bar{s}, \bar{a})\} \backslash \{(b'', s'', \bar{a})\}$, and marked $\bar{b}$ and $\bar{s}$ as inactive, b'' and s'' are reactivated.
Step 9: If there is at least one rejected pair of (b', s'), repeat Steps 5, 6, 7, 8.
Step 10: Repeat Steps 5, 6, 7, 8, 9 until each $a \in \mathcal{A}$ has her matching contract or has no more choice on pairs (b, s).
Step 11: Let $\mathcal{S}'$ be the set of stations with at least one empty spot that have not rejected any drop-off users having contracts during the period. Then, if nonvirtual drop-off users have no contract, determine their parking stations using the Serial Dictatorship mechanism. That is, they are ordered by some criteria and, in turn, select the most preferred parking station from $\mathcal{S}'$. If, after this process, there are still drop-off users who do not have a parking station, they turn to the next matching period.

By Step 11, when a drop-off agent's application for a station is rejected and he is matched with a pick-up agent, other drop-off agents are not allowed to park their cars at the rejected station during the period. This fact will satisfy fairness. Furthermore, the nature of the three-sided matching mechanism by Zhang and Zhong will ensure that stable matching is obtained within the period.

Some existing mechanisms include monetary transfer. For example, in [43], financial compensation is paid to those who take actions that reduce the imbalance. Although we do not include monetary transfer into a mechanism in this study, it is possible to pay compensation to drop-off agents who change their drop-off station as long as they do not significantly change the preference order.

10.4.4 Experiment Settings and Simulation Process

We constructed a one-way car-sharing simulator to evaluate the effectiveness and characteristics of our mechanism. Our mechanism was compared with an important previous regulation mechanism and non-regulation scenario. The regulation mechanism by [43] was compared as a baseline with our proposal, although Singla's mechanism includes monetary transfer among pick-up agents, drop-off agents, and the operator. Non-regulation scenario means that all drop-off users go to their intended drop-off locations, and pick-up agents use cars parked nearest their current locations if doing so offers a non-negative utility.

10.4.4.1 Experiment Settings

We follow the simulation setting in [45]. We configured the parameters of the one-way car-sharing simulator by referring to the literature [19, 50]. We used the following experiment settings. A service area is a rectangular area 8 km in width and 16 km in length with 13,041 nodes and 25,840 links. To prevent the moving distances from having the same value for multiple routes, we fluctuate each node by a maximum of ±20 m in the x and y coordinates. Each station is located in an approx. 500 m × 500 m rectangle. Two parking spots exist in each station, i.e., $n_{s_i} = 2$ for all s_i. The walking speed is 80 m/min (4.8 km/h). The driving speed is 500 m/min (30 km/h). For pick-up agents, we calculate the larger value of (a) the distance between the pick-up and the current locations divided by the walking speed and (b) the distance between the negotiated drop-off location and the drop-off agent's current location divided by the driving speed. For drop-off agents, we calculate (i) the distance between the negotiated drop-off and current locations divided by the driving speed + (ii) the distance between the negotiated drop-off and initial drop-off locations divided by the walking speed – (iii) the distance between the initial drop-off and current locations divided by the driving speed. Vehicles are scattered uniformly at the beginning. The number of vehicles n_v is set to 50 or 150. Demand occurs in intervals of 30 s. The batch size n_{ba} is set to 5, 10, or 15.

We referred to [19] and constructed reasonable utility functions. The utility function of pick-up agent a_t is composed of the positive valuation v_{a_t} obtained by the trip and the disutility obtained by the time x required to pick up a vehicle. Here, we assume that all vehicles are of the same type.

$$u_{a_t}(x) = v_{a_t} - 0.0001\alpha_{a_t}x^4 \tag{10.3}$$

$$(v_{a_t}, \alpha_{a_t}) \in \{(6, 6), (5, 5), (4, 4), (3, 3), (2, 2), (1, 1)\}$$

This function means that the pick-up agent can get a parked car at most 10 mins away. One of the above six different parameter values will be assigned to each agent in accordance with a uniform distribution. On the basis of studies such as [25], we assumed that drop-off agent b_j's disutility function can be formulated as follows.

$$u_{b_j}(x) = -0.21x \tag{10.4}$$

10.4.4.2 Metrics

We evaluate the proposed mechanism regarding service levels, social surplus, and drop-off agents' disutilities. Here, the service level is the ratio of the number of users who succeed in picking up a vehicle to the number of users who apply for usage, which is defined as follows.

$$\textit{Service level} = \frac{\textit{Potential customers} - \textit{No-service events}}{\textit{Potential customers}} \tag{10.5}$$

The social surplus is defined as the total utilities of the agents obtained through the service process. It can be denoted as below.

$$\textit{Social surplus} = (V_a - W_a) - D_b \tag{10.6}$$

We denoted the cumulative positive valuations obtained by the pick-up agents for the trips as V_a, the cumulative disutilities suffered by the pick-up agents for the time required to pick up vehicles $-W_a$ and the cumulative disutilities suffered by the drop-off agents for the delay in reaching the initially intended destination $-D_b$.

10.4.4.3 Simulation Process

At the beginning of the process, all vehicles are parked in the service area following a uniform distribution. Then, the system operator takes an application from a pick-up agent. According to the batch size, the matching mechanism is executed. The iteration process finishes when the system operator takes a certain number of usage applications.

10.4.5 Results

Table 10.1 shows the average service levels, social surplus, and drop-off agents' disutilities of 4,320 applications over 10 simulations under the different regulation mechanisms. The standard deviation is shown in parentheses. Our mechanism achieved higher service levels and a higher social surplus.

Table 10.1 Simulation results. The standard deviation is shown in parentheses

	50 vehicles			150 vehicles		
	Service levels	Social surplus	Disutility	Service levels	Social surplus	Disutility
No regulation	0.300 (0.0043)	3,077 (59)	0	0.651 (0.0054)	7,380 (92)	0
Baseline	0.304 (0.0038)	3,111 (92)	–44 (8)	0.654 (0.0100)	7,159 (147)	–486 (14)
Batch 05	0.329 (0.0058)	3,167 (78)	–194 (26)	0.677 (0.0063)	**7,439** (110)	–216 (21)
Batch 10	0.356 (0.0076)	**3,200** (78)	–420 (24)	0.690 (0.0100)	7,270 (144)	–489 (26)
Batch 15	**0.374** (0.0065)	3,159 (76)	–609 (27)	**0.694** (0.0107)	6,929 (149)	–788 (50)

10.4.6 Conclusions

We proposed a novel regulation mechanism based on a matching among drop-off agents, stations, and pick-up agents to reduce the imbalance between vehicle distribution and demand in station-based one-way car sharing. Our mechanism introduced three-sided matching to ask drop-off agents to change their drop-off stations to neighboring stations. We evaluated our mechanism with the baseline and a non-regulation scenario in a one-way car-sharing simulator. The experimental results showed that the proposed mechanism achieved higher service levels than those used for comparison.

References

1. D.J. Abraham, A. Blum, T. Sandholm, Clearing algorithms for barter exchange markets: Enabling nationwide kidney exchanges, in *Proceedings of the 8th ACM Conference on Electronic Commerce* (2007), pp. 295–304
2. A.R. Ahmed, M.S. Rahman, S.G. Kobourov, Online facility assignment. Theoret. Comput. Sci. **806**, 455–467 (2020)
3. A. Antoniadis, C. Fischer, A. Tönnis, A collection of lower bounds for online matching on the line, in *LATIN 2018: Theoretical Informatics* (2018), pp. 52–65
4. C. Borgs, J. Chayes, N. Immorlica, M. Mahdian, A. Saberi, Multi-unit auctions with budget-constrained bidders, in *Proceedings of the 6th ACM Conference on Electronic Commerce* (2005), pp. 44–51
5. F. Ciari, M. Balac, M. Balmer, Modelling the effect of different pricing schemes on free-floating carsharing travel demand: a test case for Zurich, Switzerland. Transportation **42**(3), 413–433 (2015)
6. L.E. Dubins, D.A. Freedman, Machiavelli and the Gale-Shapley algorithm. Am. Math. Mon. **88**(7), 485–494 (1981)
7. L. Ehlers, I.E. Hafalir, M.B. Yenmez, M.A. Yildirim, School choice with controlled choice constraints: hard bounds versus soft bounds. J. Econ. Theory **153**, 648–683 (2014)

8. A.D. Febbraro, N. Sacco, M. Saeednia, One-way carsharing: solving the relocation problem. Transp. Res. Rec. **2319**(1), 113–120 (2012)
9. D. Fragiadakis, A. Iwasaki, P. Troyan, S. Ueda, M. Yokoo, Strategyproof matching with minimum quotas. ACM Trans. Econ. Comput. **4**(1), 6:1–6:40 (2016)
10. B. Fuchs, W. Hochstättler, W. Kern, Online matching on a line. Theoret. Comput. Sci. **332**(1–3), 251–264 (2005)
11. D. Gale, L.S. Shapley, College admissions and the stability of marriage. Am. Math. Mon. **69**(1), 9–15 (1962)
12. D. Gavalas, C. Konstantopoulos, G. Pantziou, Chapter 13-design and management of vehicle-sharing systems: a survey of algorithmic approaches, in *Smart Cities and Homes* (Morgan Kaufmann, 2016), pp. 261–289
13. A.V. Goldberg, J.D. Hartline, A. Wright, Competitive auctions and digital goods, in *Proceedings of the 12th Annual Symposium on Discrete Algorithms* (2001), pp. 735–744
14. M. Goto, A. Iwasaki, Y. Kawasaki, R. Kurata, Y. Yasuda, M. Yokoo, Strategyproof matching with regional minimum and maximum quotas. Artif. Intell. **235**, 40–57 (2016)
15. I.E. Hafalir, M.B. Yenmez, M.A. Yildirim, Effective affirmative action in school choice. Theor. Econ. **8**(2), 325–363 (2013)
16. N. Hamada, C.-L. Hsu, R. Kurata, T. Suzuki, S. Ueda, M. Yokoo, Strategy-proof school choice mechanisms with minimum quotas and initial endowments. Artif. Intell. **249**, 47–71 (2017)
17. T. Harada, T. Itoh, S. Miyazaki, Capacity-insensitive algorithms for online facility assignment problems on a line (2022). arXiv:2207.05308
18. J.W. Hatfield, P.R. Milgrom, Matching with contracts. Am. Econ. Rev. **95**(4), 913–935 (2005)
19. S. Herrmann, F. Schulte, S. Voß, Increasing acceptance of free-floating car sharing systems using smart relocation strategies: a survey based study of car2go Hamburg, in *Computational Logistics* (2014), pp. 151–162
20. T. Itoh, S. Miyazaki, M. Satake, Competitive analysis for two variants of online metric matching problem. Discrete Math. Algorithms Appl. **13**(6), 2150156:1–2150156:16 (2021)
21. D. Jorge, G. Molnar, G.H. de Almeida Correia, Trip pricing of one-way station-based carsharing networks with zone and time of day price variations. Transp. Res. Part B: Methodol. **81**, 461–482 (2015)
22. B. Kalyanasundaram, K. Pruhs, Online weighted matching. J. Algorithms **14**(3), 478–488 (1993)
23. B. Kalyanasundaram, K. Pruhs, On-line network optimization problems, in *Online Algorithms: The State of the Art* (Springer, 1996), pp. 268–280
24. Y. Kamada, F. Kojima, Efficient matching under distributional constraints: theory and applications. Am. Econ. Rev. **105**(1), 67–99 (2015)
25. H. Kato, A. Sakashita, T. Tsuchiya, Web-based versus paper-based survey data: An estimation of road users' value of travel time savings. J. East. Asia Soc. Transp. Stud. **11**, 2502–2518 (2015)
26. S. Khuller, S.G. Mitchell, V.V. Vazirani, On-line algorithms for weighted bipartite matching and stable marriages. Theoret. Comput. Sci. **127**(2), 255–267 (1994)
27. F. Kojima, School choice: impossibilities for affirmative action. Games Econ. Behav. **75**(2), 685–693 (2012)
28. F. Kojima, A. Tamura, M. Yokoo, Designing matching mechanisms under constraints: an approach from discrete convex analysis. J. Econ. Theory **176**, 803–833 (2018)
29. B. Korte, J. Vygen, *Combinatorial Optimization: Theory and Algorithms* (Springer, 2018)
30. E. Koutsoupias, A. Nanavati, The online matching problem on a line, in *Proceedings of the 1st International Workshop on Approximation and Online Algorithms* (Springer, 2003), pp. 179–191
31. R. Kurata, N. Hamada, A. Iwasaki, M. Yokoo, Controlled school choice with soft bounds and overlapping types. J. Artif. Intell. Res. **58**, 153–184 (2017)
32. K. Liu, K. Yahiro, M. Yokoo, Strategyproof mechanism for two-sided matching with resource allocation. Artif. Intell. **316**, 103855 (2023)

33. M. Maciejewski, J. Bischoff, K. Nagel, An assignment-based approach to efficient real-time city-scale taxi dispatching. IEEE Intell. Syst. **31**(1), 68–77 (2016)
34. K. Nayyar, S. Raghvendra, An input sensitive online algorithm for the metric bipartite matching problem, in *Proceedings of the 58th IEEE Annual Symposium on Foundations of Computer Science* (2017), pp. 505–515
35. N. Nisan, T. Roughgarden, E. Tardos, V.V. Vazirani, *Algorithmic Game Theory* (Cambridge University Press, 2007)
36. L. Pan, Q. Cai, Z. Fang, P. Tang, L. Huang, A deep reinforcement learning framework for rebalancing dockless bike sharing systems, in *Proceedings of the 33rd AAAI Conference on Artificial Intelligence* (2019), pp. 1393–1400
37. E. Peserico, M. Scquizzato, Matching on the line admits no $o(\sqrt{\log n})$-competitive algorithm, in *Proceedings of the 48th International Colloquium on Automata, Languages, and Programming* (2021), pp. 103:1–103:3
38. S. Raghvendra, A robust and optimal online algorithm for minimum metric bipartite matching, in *Approximation, Randomization and Combinatorial Optimization: Algorithms and Techniques* (2016), pp. 18:1–18:16
39. S. Raghvendra, Optimal analysis of an online algorithm for the bipartite matching problem on a line, in *Proceedings of the 34th International Symposium on Computational Geometry* (2018), pp. 67:1–67:14
40. A.E. Roth, The economics of matching: stability and incentives. Math. Oper. Res. **7**(4), 617–628 (1982)
41. A.E. Roth, M.A.O. Sotomayor, *Two-Sided Matching: A Study in Game-Theoretic Modeling and Analysis* (Cambridge University Press, 1990)
42. S. Schmöller, S. Weikl, J. Müller, K. Bogenberger, Empirical analysis of free-floating carsharing usage: the Munich and Berlin case. Transp. Res. Part C: Emerging Technol. **56**, 34–51 (2015)
43. A. Singla, M. Santoni, G. Bartók, P. Mukerji, M. Meenen, A. Krause, Incentivizing users for balancing bike sharing systems, in *Proceedings of the 29th AAAI Conference on Artificial Intelligence* (2015), pp. 723–729
44. D.D. Sleator, R.E. Tarjan, Amortized efficiency of list update and paging rules. Commun. ACM **28**(2), 202–208 (1985)
45. K. Takahira, S. Matsubara, Contract-based inter-user usage coordination in free-floating car sharing, in *Proceedings of the 35th AAAI Conference on Artificial Intelligence* (2021), pp. 11361–11368
46. K. Tomoeda, Finding a stable matching under type-specific minimum quotas. J. Econ. Theory **176**, 81–117 (2018)
47. X. Wang, N. Agatz, A. Erera, Stable matching for dynamic ride-sharing systems. Transp. Sci. **52**(4), 850–867 (2018)
48. A. Waserhole, V. Jost, N. Brauner, Pricing techniques for self regulation in vehicle sharing systems. Electron. Notes Discrete Math. **41**, 149–156 (2013)
49. S. Weikl, K. Bogenberger, Relocation strategies and algorithms for free-floating car sharing systems. IEEE Intell. Transp. Syst. Mag. **5**(4), 100–111 (2013)
50. G. Wielinski, M. Trépanier, C. Morency, What about free-floating carsharing? Transp. Res. Rec. **2536**(1), 28–36 (2015)
51. F. Zhang, L. Zhong, Three-sided matching problem with mixed preferences. J. Comb. Optim. **42**(4), 928–936 (2021)
52. H. Zhang, J. Zhao, Mobility sharing as a preference matching problem. IEEE Trans. Intell. Transp. Syst. **20**(7), 2584–2592 (2019)