The Dissertation Committee for Fu Li
certifies that this is the approved version of the following dissertation:

Topics in Computational Social Choice Theory

Committee:

Greg Plaxton, Supervisor

David Zuckerman

Shuchi Chawla

Vijay K. Garg

Topics in Computational Social Choice Theory

by

Fu Li

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

May 2022

Acknowledgments

I would like to express my deep gratitude to my supervisor, Prof. Greg Plaxton, for the continuous support of my Ph.D study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all of the stages of my dissertation work. He has provided me with not only a lot of useful advice, but also the freedom to pursue various interesting ideas.

I would also like to thank the other members of my dissertation committee, Prof. David Zuckerman, Prof. Shuchi Chawla, and Prof. Vijay K. Garg, for their patience and feedback.

I would like to thank my collaborators, Vaibhav Sinha and Xiong Zheng, for our many stimulating discussions. Last but not least, I would like to thank my family, and my friends, for their support and encouragement.

Topics in Computational Social Choice Theory

Publication No. _____

Fu Li, Ph.D.
The University of Texas at Austin, 2022

Supervisor: Greg Plaxton

Collective decision making arises in many real life situations, such as elections, matching applicants to job openings, and partitioning students into working groups. Over the past three decades, computer scientists have used techniques from the design and analysis of algorithms and complexity theory to develop social choice theory from a computational perspective, leading to the new research area of computational social choice theory. Broadly speaking, this dissertation makes two kinds of contributions to this new research area.

First, we provide new algorithms and hardness results for two settings. In our first setting, we revisit the classical housing markets model in which we seek to compute a suitable reallocation of n houses to n agents, each of whom initially owns a particular house and has strict preferences over the set of n houses. In the variants of this model that we study, graph-based restrictions are imposed on the exchange of houses. More specifically, we consider two cases. In the first case, we are given a graph where the vertex set corresponds to the set of agents, and we assume that two agents can only swap houses if they are adjacent in the graph and the swap is Pareto-improving; thus, in this case, the locations of the agents remain fixed in the graph, while the houses can move around. The second case is similar except we assume that the locations of the houses remain fixed (i.e., each vertex now corresponds to a house) and the agents move around. In our second setting, we consider a variant of the classical coalition formation game, the fractional hedonic game. Such a game can be represented by a directed weighted graph where the set of vertices corresponds to the set of players, and where the weight of an edge

(i, j) from player i to player j denotes the value that player i has for player j. Given a partitioning of the players into coalitions, the utility of player i is defined to be the average value that player i assigns to the members of their coalition. We assume that there is a limit on the number of coalitions that can be formed. In this setting, we study the problem of computing a social welfare maximizing partition and the problem of computing a Nash stable partition. We study these two problems for a variety of different graph classes.

Second, we design mechanisms without money for new games related to facility location and resource sharing. In the facility location game that we consider, a central planner plans to build a heterogeneous set of facilities (e.g., a school, a water treatment plant) and each agent reports which of these facilities they consider to be "obnoxious". The location of each agent is assumed to be fixed, and the utility of an agent is based on the minimum distance of the agent to an obnoxious facility. The goal is to design strategyproof mechanisms that maximize either the sum of the agent utilities or the minimum utility of any agent. In the resource sharing game that we consider, agents pool their computational resources in order to better accommodate fluctuations in individual demand over a sequence of rounds, and the agents specify their demand for each round at the outset. We present a group-strategyproof and non-wasteful mechanism that guarantees each agent at least half of the utility they could achieve without sharing their resources.

Table of Contents

List of Tables

Chapter 1

Introduction

Collective decision making arises in many real-life situations, such as political election, assigning positions to applicants, and partitioning students into working groups. In such situations, we are given a set of agents with individual and possibly conflicting preferences, and we seek to aggregate their preferences to reach a satisfactory collective decisions. This is the fundamental challenge that has given rise to the field of social choice theory. Social choice theory dates back to Condorcet's voting paradox in the 18th century, and was significantly influenced by the seminal work of Kenneth Arrow in the 1950s. Arrow proved that there does not exist a reasonable preference aggregation rule that satisfies a small set of basic requirements [13]. To prove this, Arrow introduced the axiomatic method into the study of aggregation methods Since then, much of the work in social choice theory has used axiomatic methods to study the formal possibility and impossibility of aggregation methods that achieve certain combinations of desirable properties. We refer to a two-volume book by Arrow, Sen, and Suzumura [14, 15] for a detailed reference on classical social choice theory.

Over the past few decades, computer scientists have developed social

choice theory from a computational perspective, leading to the new research area of computational social choice theory. There are at least two reasons for this development. First, the practical utility of an aggregation method depends not only on the game-theoretic properties it provides, but also on its implementation complexity. The field of computer science offers powerful tools, including algorithm design, complexity theory, and communication complexity, for analyzing the implementation complexity of an aggregation method. Computer scientists have also identified new applications for social choice theory. One example of such an application is Internet search. A preference aggregation method can be used to aggregate the output of several search engines that rank web pages by relevance. Google's PageRank algorithm uses this idea and has been analyzed in terms of the axiomatic method [140]. Another important application is associated with cloud computing. The cloud can be viewed as a new paradigm for delivering computing as a utility, and it is often impossible for cloud providers to satisfy all of the demands of their users. Under such overloaded condition, it is important to allocate cloud resources fairly. Many researchers have used social choice theory to systematically study various cloud resource allocation mechanisms. These new application areas have inspired researchers to propose many new problems, and to consider social choice from new perspectives. For a detailed survey of computational social choice theory, we refer the reader to the book by Brandt et al. [44].

A lot of the current research in computational social choice theory is motivated by emerging applications in the modern era of networked communi-

cation and big data. The Internet enables agents (e.g., computers or people) all over the world to communicate with one another at high speed. This encourages agents to engage with ever larger and more widespread groups. These groups of agents often make collective decisions subject to both external and internal constraints; relevant examples include kidney exchange [64], matching students to seats in schools [2], and distributing food to charities [8]. The Internet has also enabled the collection of agent-specific data on a massive scale. For certain applications, this data makes it easier than ever before to infer agent preferences. However, to capture the full potential of such data, more efficient algorithms are needed for collecting and processing agent-specific data.

Mechanisms are often partitioned into two broad categories: those based on money, and those that do not involve money. An example of a mechanism based on money is the celebrated Vickrey-Clarke-Groves (VCG) mechanism for combinatorial auctions [163, 58, 92]. One strength of the VCG mechanism is that it follows an intuitively clear rule: It charges each agent the harm that they cause to other agents. A second strength of the VCG mechanism is that it satisfies a property called strategyproofness: no agent benefits by misreporting their preferences, regardless of the reports of the other agents. Mechanisms without money are motivated by certain real-world applications in which the usage of money is undesirable. For example, monetary transfers may be considered unethical or illegal in the context of kidney exchange, or the assignment of students to seats in public schools.

This dissertation focuses on mechanisms without money, and studies both the design and the computational complexity of such mechanisms. In all of the settings that we address, a set of self-interested agents formulate and report preferences to a (centralized or decentralized) coordination mechanism, which uses the agent preferences to allocate (or reallocate) a limited number of resources. The coordination mechanism chooses outcomes in order to achieve one or more social objectives, such as strategyproofness or Pareto-optimality.

In particular, this dissertation focuses primarily on questions related to the following four topics in computational social choice theory: housing markets, hedonic games, facility location games, and resource sharing. Below we briefly introduce each of these topics.

1.1 Housing Markets

Resource allocation is the process of distributing a set of resources among a group of agents, such that the outcome is efficient for society and reasonably fair to each agent. Problems related to resource allocation under preferences are widely studied in both computer science and economics. Research in this area seeks to gain mathematical insight into the structure of resource allocation problems, and to exploit this structure to design procedures that satisfy some guarantee of efficiency or fairness. For example, envy-freeness is a natural fairness criterion requiring that no agent prefers the share assigned to another agent to the own share. Regarding efficiency, a basic requirement is Pareto-efficiency, imposing that it is impossible to improve the

satisfaction of some agents without harming another.

In one important class of resource allocation problems, sometimes referred to as one-sided matching problems [125], we seek to allocate indivisible objects to a set of agents, where each agent has preferences over the objects and wants to receive at most one object (unit demand). The allocation should enjoy one or more strong game-theoretic properties, such as Pareto-efficiency.

In a seminal work, Shapley and Scarf [154] introduced the notion of a housing market, which corresponds to the special case of one-sided matching in which there are an equal number of agents and objects, each agent is initially endowed with a distinct object, and each agent is required to be matched to exactly one object. They present an elegant algorithm (attributed to David Gale) for housing markets called the top trading cycles (TTC) algorithm. The TTC algorithm enjoys a number of strong game-theoretic properties. For example, when agents have strict preferences, the output of the TTC algorithm is the unique matching in the core. It has subsequently been generalized to handle more complex variants of the original housing market problem (e.g., [16, 57, 73, 149, 152]).

1.2 Hedonic Games

Coalition formation is a field analyzing one or more groups of agents, called coalitions, that get together to jointly determine their actions. Hedonic games are a notable type of games on studying coalition formation (see [22] for a survey). A hedonic game is specified by a set of players who have preferences

over the set of all possible players' coalitions. The outcome of a hedonic game consists a partition of the players into disjoint coalitions. The main requirement in hedonic games is the stability of the solution. Different notions of stability can be established. The basic one is Nash stability where no agent would prefer to belong to another group.

1.3 Facility Location Games

Facility location problem is a classical commerce optimization problem. A facility may be an outlet, workstation, hospital, and school. In facility location problem, with reported locations or preference of agents, an authority needs to decide where to place the facilities for minimizing social cost. Later on, Procaccia and Tannenholtz introduced a game-theoretic version of this problem which is known as the facility location game [147]. In facility location games, each agent is self-interested, i.e., who may misreport in order to change the outcome of the construction sites and hence benefit from the outcome. Facility location games aim to design mechanisms that minimize the social cost on the condition that agents will not gain by misreporting their information.

1.4 Resource Sharing

The proliferation of Internet-based technologies has allowed sharing-related applications to flourish. As the "sharing economy" continues to expand, it will be important to develop clear rules to support the true joint ownership of scarce resources. As devices are increasingly connected and con-

trolled by algorithms, an attractive option is to use algorithms to formally encode such rules. For example, Freeman et al. [83] investigated a sharing-related setting in which agents pool their computational resources to improve utilization and efficiency. In this setting, the authors strive to develop mechanisms that incentive sharing and achieve other desirable properties such as nonwastefulness and strategyproofness. For a dynamic version of this setting, it is shown to be impossible to simultaneously maintain all of these properties; consequently, trade-offs are investigated. For more details on algorithmically-driven shared ownership economies, we refer to the article by Conitzer and Freeman [60].

1.5 Organization of the Rest of the Dissertation

This section gives a brief overview of our contributions. We remark that the results presented in Chapters 2, 5, and 6 are based on joint work with Greg Plaxton and Vaibhav Sinha [119, 117, 118], the results presented in Chapter 3 are based on joint work with Xiong Zheng [120], and the results presented in Chapter 4 are based on a single-authored paper [116]. In order to minimize duplication, the results coauthored with Vaibhav Sinha have been partititoned between his thesis [156] and this disseration. This partitioning is discussed in greater detail in the relevant chapters.

1.5.1 Results Related to Housing Markets

In recent work, Gourvès, Lesca, and Wilczynski propose a variant of the classic housing markets model where the matching between agents and objects evolves through Pareto-improving swaps between pairs of adjacent agents in a social network. To explore the swap dynamics of their model, they pose several basic questions concerning the set of reachable matchings. In their work and other follow-up works, these questions have been studied for various classes of graphs: stars, paths, generalized stars (i.e., trees where at most one vertex has degree greater than two), trees, and cliques. For generalized stars and trees, it remains open whether a Pareto-efficient reachable matching can be found in polynomial time.

We first pursue the same set of questions under a natural variant of their model. In our model, the social network is replaced by a network of objects, and a swap is allowed to take place between two agents if it is Pareto-improving and the associated objects are adjacent in the network. In those cases where the question of polynomial-time solvability versus NP-hardness has been resolved for the social network model, we are able to show that the same result holds for the network-of-objects model. In addition, for our model, we present a polynomial-time algorithm for computing a Pareto-efficient reachable matching in generalized star networks. Moreover, the object reachability algorithm that we present for path networks is significantly faster than the known polynomial-time algorithms for the same question in the social network model. These results are presented in details in Chapter 2.

We next study how to direct the agents to swap objects with each other in order to arrive at a reachable matching that is both efficient and most agreeable in the social network model. In particular, we study the computational complexity of reaching a Pareto-efficient matching that maximizes the number of agents who prefer their match to their initial endowments. We consider the same graph structures of the social network studied before: stars, paths, generalized stars, trees, and cliques. Additionally, we consider two assumptions regarding preference relations of agents: strict (ties among objects not allowed) or weak (ties among objects allowed). By designing two polynomial-time algorithms and two NP-hardness reductions, we resolve the complexity of all cases not yet known. Our main contributions include a polynomial-time algorithm for path networks with strict preferences and an NP-hardness result in a star network with weak preferences.

1.5.2 Results Related to Hedonic Games

Recently, fractional hedonic games have received considerable attention. Such a game can be represented by directed weighted graph where the weight of edge (i, j) denotes the value player i has for player j. The utility of player i is the average value that player i assigns to the members of i's coalition. We study a variant of this game where there is a specific upper bound k on the number of coalitions that can be formed. We first consider how to find a coalition partition that maximizes the social welfare, i.e., the sum of the utilities. Computing social welfare maximizing partitions for these games of all agents

on undirected unweighted graphs is known to be NP-hard. Here, we study the parameterized complexity in terms of k. For all fixed $k \geq 2$, we show that it remains NP-hard to find a social welfare maximizing k-partition for undirected unweighted graphs. For undirected unweighted trees, we present an algorithm finding a social welfare maximizing k-partition in polynomial time. Moreover, we consider Nash stable outcomes. We show that for all $k \geq 2$, if a fractional hedonic game on a directed unweighted graph with bounded maximum out-degree admits a Nash stable k-partition, then the stable partition is almost balanced. However, we prove that determining whether a fractional hedonic game admits a Nash stable k-partition is NP-complete for all $k \geq 2$. These results are presented in details in Chapter 4.

1.5.3 Results Related to Facility Location Games

We consider a facility location game in which n agents reside at known locations on a path, and k heterogeneous facilities are to be constructed on the path. Each agent is adversely affected by some subset of the facilities, and is unaffected by the others. We design two classes of mechanisms for choosing the facility locations given the reported agent preferences: utilitarian mechanisms that strive to maximize social welfare (i.e., to be efficient), and egalitarian mechanisms that strive to maximize the minimum welfare. For the utilitarian objective, we present a weakly group-strategyproof efficient mechanism for up to three facilities, we give a strongly group-strategyproof mechanism that guarantees at least half of the optimal social welfare for arbitrary k, and we

prove that no strongly group-strategyproof mechanism achieves an approximation ratio of 5/4 for one facility. For the egalitarian objective, we present a strategyproof egalitarian mechanism for arbitrary k, and we prove that no weakly group-strategyproof mechanism achieves a $o(\sqrt{n})$ approximation ratio for two facilities. We extend our egalitarian results to the case where the agents are located on a cycle, and we extend our first egalitarian result to the case where the agents are located in the unit square. These results are presented in details in Chapter 5.

1.5.4 Results Related to Resource Sharing

It is often beneficial for agents to pool their resources in order to better accommodate fluctuations in individual demand. Many multi-round resource allocation mechanisms operate in an online manner: in each round, the agents specify their demands for that round, and the mechanism determines a corresponding allocation. In this dissertation, we focus instead on the offline setting in which the agents specify their demand for each round at the outset. We formulate a specific resource allocation problem in this setting, and design and analyze an associated mechanism based on the solution concept of lexicographic maximin fairness. We present an efficient implementation of our mechanism, and prove that it is Pareto-efficient, envy-free, non-wasteful, resource monotonic, population monotonic, and group strategyproof. We also prove that our mechanism guarantees each agent at least half of the utility that they can obtain by not sharing their resources. We complement these

positive results by proving that no maximin fair mechanism can improve on the aforementioned factor of one-half. These results are presented in details in Chapter 6.

Chapter 2

Object Allocation Over a Network of Objects

2.1 Introduction

Problems related to resource allocation under preferences are widely studied in both computer science and economics. Research in this area seeks to gain mathematical insight into the structure of resource allocation problems, and to exploit this structure to design fast algorithms. In one important class of resource allocation problems, sometimes referred to as one-sided matching problems [125], we seek to allocate indivisible objects to a set of agents, where each agent has preferences over the objects and wants to receive at most one object (unit demand). The allocation should enjoy one or more strong game-theoretic properties, such as Pareto-efficiency.

In a seminal work, Shapley and Scarf [154] introduced the notion of a housing market, which corresponds to the special case of one-sided matching in which there are an equal number of agents and objects, each agent is initially endowed with a distinct object, and each agent is required to be matched to

A part of the contents of this chapter appeared in [119]. The hardness results presented in Section 2.5 are primarily due to Vaibhav Sinha, and are included in his M.Sc. thesis; these results are included here for the sake of completeness. Except these hardness results, this chapter presents the results for which I was the primary contributor.

exactly one object. They present an elegant algorithm (attributed to David Gale) for housing markets called the top trading cycles (TTC) algorithm. The TTC algorithm enjoys a number of strong game-theoretic properties. For example, when agents have strict preferences, the output of the TTC algorithm is the unique matching in the core. The TTC algorithm has subsequently been generalized to handle more complex variants of the original housing market problem (e.g., [16, 73, 149]).

Like many one-sided matching algorithms, the TTC algorithm is centralized: it takes all of the agent preference information as input and computes the output matching. In some resource allocation scenarios of practical interest, it may be difficult or impossible to coordinate such a global recomputation of the matching. Accordingly, researchers have studied decentralized (or distributed) variants of one-sided matching problems in which the initial allocation gradually evolves as "local" trading opportunities arise. In this setting, restrictions are imposed on the sets of agents that are allowed to participate in a single trade. For example, we might only allow (certain) pairs of agents to trade. In addition, all trades are required to be Pareto-improving. Locality-based restrictions on trade are generally enforced through graph-theoretic constraints.

Of particular relevance to the present chapter is the line of research initiated by Gourvès et al. [91] on decentralized allocation in housing markets. They propose a model in which agents have strict preferences and are embedded in an underlying social network. A pair of agents are allowed to

swap objects with each other only if (1) they will be better off after the swap, and (2) they are directly connected (socially tied) via the network. The underlying social network is modeled as an undirected graph, and five different graph classes are considered: paths, stars, generalized stars, trees, and general graphs. The swap dynamics of the model are investigated by considering three computational questions. The first question, Reachable Object, asks whether there is a sequence of swaps that results in a given agent being matched to a given target object. The second question, Reachable Matching, asks whether there is a sequence of swaps that results in a given target matching. The third question, Pareto Efficiency, asks how to find a sequence of swaps that results in a Pareto-efficient matching with respect to the set of reachable matchings.

Gourvès et al. [91] studied each of the three questions in the context of the aforementioned graph classes, with the goal of either exhibiting a polynomial-time algorithm or establishing NP-hardness. For some of these problems, it is a relatively straightforward exercise to design a polynomial-time algorithm (even for the search version). In particular, this is the case for all three reachability questions on stars, for Pareto Efficiency on paths, and for Reachable Matching on trees (which subsumes Reachable Matching on generalized stars, and hence also on paths). Gourvès et al. present an elegant reduction from 2P1N-SAT [167] to establish the NP-completeness of Reachable Object on generalized stars (and hence also on trees and general graphs). They establish the NP-completeness of Reachable Matching on general graphs via a reduction from Reachable Object on trees. The latter reduction has the

property that for any given instance of Reachable Object on trees, the target matching in the instance of Reachable Matching on general graphs produced by the transformation matches each agent to its most preferred object. Consequently, the same reduction establishes the NP-hardness of Pareto Efficiency on general graphs. The work of Gourvès et al. left three of these problems open: Reachable Object on paths and Pareto Efficiency on generalized stars and trees. Subsequently, two sets of authors independently presented polynomial-time algorithms for Reachable Object on paths [28, 95]. Both groups obtained an $O(n^4)$-time algorithm by carefully studying the structure of swap dynamics on paths and then reducing the problem to 2-SAT. The complexity of Pareto Efficiency remains open for generalized stars and for trees. Gourvès et al. noted, "It appears interesting to see if Pareto (Efficiency) is polynomial time solvable in a generalized star by a combination of the techniques used to solve the cases of paths and stars."

Bentert et al. [28] established that Reachable Object on cliques is NP-complete, and Müller and Bentert [130] established that Reachable Matching on cliques is NP-complete. It is easy to extend the latter result to show that Pareto Efficiency on cliques is NP-hard. These three hardness results for cliques subsume the corresponding results obtained previously for general graphs by Gourvès et al.

We study a natural variant of the decentralized housing markets model of Gourvès et al. [91]. Instead of enforcing locality constraints on trade via a network where the locations of the agents are fixed (since they correspond to

the vertices of the network) and the objects move around (due to swaps), we consider a network where the locations of the objects are fixed and the agents move around. We refer to these two models as the object-moving model and the agent-moving model. Table 2.1 summarizes the current state of the art for the object-moving model.

To motivate the study of the agent-moving model, consider a cloud computing environment with a large number of servers (objects) connected by a network that are available to rent. A set of customers (agents) are each interested in renting one server. The servers vary in CPU capacity, storage capacity, physical security, and rental cost. Varying customer workloads and requirements result in varying customer preferences over the servers. Rather than attempting to globally optimize the entire matching of customers to servers, it might be preferable to allow local swaps between adjacent servers to gradually optimize the matching. Given that customer workloads are likely to vary significantly over time, an optimization strategy based on frequent local updates might outperform a strategy based on less frequent global updates. Alternatively, one can envision a system that performs occasional global updates to optimize the matching, and that relies on local updates to maintain a reasonable matching between successive global updates.

Our Results. We initiate the study of the agent-moving model by revisiting each of the questions associated with Table 2.1 in the context of the agent-moving model. We emphasize that the sole difference between the agent-moving model and the object-moving model is that the locality constraint pre-

	Reachable Object	Reachable Matching	Pareto Efficiency
Star	poly-time	(poly-time)	poly-time
Path	poly-time	(poly-time)	poly-time
Generalized Star	NP-complete	(poly-time)	open
Tree	(NP-complete)	poly-time	open
Clique	NP-complete	NP-complete	NP-hard

Table 2.1: This table presents known complexity results for various questions related to the object-moving model of Gourvès et al. [91]. The results in parentheses follow directly from other table entries. For the agent-moving model, we obtain the same results, except that we also give a polynomial-time algorithm for Pareto Efficiency on generalized stars.

vents an agent a currently matched to an object b from trading with an agent a' currently matched to an object b' unless objects b and b' (two vertices in a given network of objects) are adjacent, rather than requiring agents a and a' (two vertices in a given network of agents) to be adjacent. Both models also require swaps to be Pareto-improving. The two models have strong similarities. In fact, for all of the questions in Table 1 for which a polynomial-time algorithm or hardness result has been established in the object-moving model, we establish a corresponding result in the agent-moving model. Moreover, for Pareto Efficiency on generalized stars, which is open in the object-moving model, we provide a polynomial-time algorithm in the agent-moving model. In some cases, it is relatively straightforward to adapt known results for the object-moving model to the agent-moving model. Below we highlight our four main technical contributions, which address more challenging cases.

Our first main technical result is an $O(n^2)$ time algorithm for Reachable Object on paths in the agent-moving model, which is much faster than the

known $O(n^4)$-time algorithms for Reachable Object on paths in the object-moving model. (Here n denotes the number of agents/objects; the size of the input is quadratic in n since the preference list of each agent is of length n.) The speedup is due to a simpler local characterization of the reachable matchings on a path in the agent-moving model.

In our second main technical result, we obtain the same $O(n^2)$ time bound for Pareto Efficiency on paths. Our algorithms for Reachable Object and Pareto Efficiency are based on an efficient subroutine for solving a certain constrained reachability problem. Roughly speaking, this subroutine determines all of the possible matches for a given agent when certain agent-object pairs are required to be matched to one another. Our implementation involves a trivial $O(n^2)$-time preprocessing phase followed by an $O(n)$-time greedy phase. The preferences of the agents are only examined during the preprocessing phase. The proof of correctness of the greedy phase is somewhat nontrivial. We solve Reachable Object on paths using a single application of the subroutine, yielding an $O(n^2)$ bound. Our polynomial-time algorithm for Pareto Efficiency on paths uses n applications of our algorithm for Reachable Object on paths. Since the preprocessing phase only needs to be performed once, the overall running time remains $O(n^2)$.

In our third main technical result, we present a polynomial-time algorithm for Pareto Efficiency on generalized stars, which remains open in the object-moving model. To tackle this problem, we use the serial dictatorship algorithm with the novel idea of dynamically choosing the dictator sequence.

We also leverage our techniques for solving Pareto Efficiency on paths.

The faster time bounds discussed above for the case of paths suggest that the agent-moving model is simpler than the object-moving model, at least from an upper bound perspective. Accordingly, we can expect it to be a bit more challenging to establish the NP-completeness results stated in Table 2.1 for the agent-moving model than for the object-moving model. In our fourth main technical result, we adapt an NP-completeness proof developed by Bentert et al. [28] in the context of the object-moving model to the more challenging setting of the agent-moving model. Specifically, we modify their reduction from 2P1N-SAT to establish that Reachable Object on cliques remains NP-complete in the agent-moving model.

Related work. For the object-moving model, Huang and Xiao [94] study Reachable Object with weak preferences, i.e., where an agent can be indifferent between different objects. Bentert et al. [28] establish NP-hardness for Reachable Object on cliques, and consider the case where the preference lists have bounded length. Saffidine and Wilczynski [151] propose a variant of Reachable Object where we ask whether a given agent is guaranteed to achieve a specified level of satisfaction after any maximal sequence of rational exchanges. Müller and Bentert [130] study Reachable Matching on cliques and cycles. Aspects related to social connectivity are also addressed in recent work on envy-free allocations [30, 33] and on trade-offs between efficiency and fairness [97].

Our agent-moving model can be viewed as a game in which each agent

seeks to be matched to an object that is as high as possible on its preference list. If the game reaches a state in which no further swaps can be performed, we say that an equilibrium matching has been reached. Agarwal et al. [7] study a similar game motivated by Schelling's well-known residential segregation model. As in our game, there are an equal number of agents and objects, the objects correspond to the nodes of a graph, a matching is maintained between the agents and the objects, and the matching evolves via Pareto-improving, agent-moving swaps. There are also some significant differences. In our model, each agent has static preferences over the set of objects, and swaps can only occur between adjacent agents (i.e, agents matched to adjacent objects). In the Agarwal et al. game, each agent has a type, the desirability of an object b to an agent a depends on the current fraction of agents in the "neighborhood" of b (i.e., the set of agents matched to an object adjacent to b) with the same type as a, and swaps can occur between any pair of agents. Agarwal et al. study the existence, computational complexity, and quality of equilibrium matchings in such games. Bilò et. al [32] further investigated the influence of the graph structure on the resulting strategic multi-agent system.

Organization of the chapter. The remainder of the chapter is organized as follows. Section 2.2 provides formal definitions. Section 2.3 presents our polynomial-time algorithms for Reachable Object and Pareto Efficiency on paths. Section 2.4 presents our polynomial-time algorithm for Pareto Efficiency on generalized stars. Section 2.5 presents our NP-completeness result for Reachable Object on cliques. Section 2.6 presents our other NP-completeness

and NP-hardness results. Section 2.7 briefly discusses simple algorithms for justifying the other polynomial-time entries in Table 2.1.

The hardness results presented in Section 2.5 are primarily due to Vaibhav Sinha, and are included in his M.Sc. thesis; these results are included here for the sake of completeness.

2.2 Preliminaries

We define an object allocation framework (OAF) as a 4-tuple $F = (A, B, \succ, E)$ where A is a set of agents, B is a set of objects such that $|A| = |B|$, $\succ$ is a collection of strict linear orderings $\{\succ_a\}_{a \in A}$ over B such that $\succ_a$ specifies the preferences of agent a over B, and E is the edge set of some undirected graph (B, E).

We define a matching μ of given OAF $F = (A, B, \succ, E)$ as a subset of $A \times B$ such that no agent or object belongs to more than one pair in μ. (Put differently, μ is a matching in the complete bipartite graph of agents and objects.) We say that such a matching is perfect if $|\mu| = |A|$. For any matching μ, we define agents(μ) (resp., objects(μ)) as the set of all matched agents (resp., objects) with respect to μ. For any matching μ and any agent a that is matched in μ, we use the shorthand notation $\mu(a)$ to refer to the object matched to agent a. For any matching μ and any object b that is matched in μ, we use the notation $\mu^{-1}(b)$ to refer to the agent matched to object b.

For any OAF $F = (A, B, \succ, E)$, any perfect matching μ of F, and any

edge $e = (b, b')$ in E such that $b' \succ_a b$ and $b \succ_{a'} b'$ where $a = \mu^{-1}(b)$ and $a' = \mu^{-1}(b')$, we say that a swap operation is applicable to μ across edge e, and we write $\mu \rightarrow_{F,e} \mu'$ where

$$\mu' = (\mu \setminus \{(a, b), (a', b')\}) \cup \{(a, b'), (a', b)\},$$

is the matching of F that results from applying this operation. We write $\mu \rightarrow_F \mu'$ to denote that $\mu \rightarrow_{F,e} \mu'$ for some edge e. We write $\mu \rightsquigarrow_F \mu'$ if there exists a sequence $\mu = \mu_0, \ldots, \mu_k = \mu'$ of matchings of F such that $\mu_{i-1} \rightarrow_F \mu_i$ for $1 \leq i \leq k$.

We define a configuration as a pair $\chi = (F, \mu)$ where F is an OAF and μ is a perfect matching of F.

For any configuration $\chi = (F, \mu)$ where $F = (A, B, \succ, E)$, any agent a in A, and any object b in B, we define $\chi(a)$ as a shorthand for the object $\mu(a)$, and we define $\chi^{-1}(b)$ as a shorthand for the agent $\mu^{-1}(b)$.

For any configuration $\chi = (F, \mu)$ where $F = (A, B, \succ, E)$, and any matching μ' of F such that $\mu \rightarrow_{F,e} \mu'$ for some edge e in E, we say that a swap is applicable to χ across edge e, and the result of applying this operation is the configuration (F, μ').

For any configuration $\chi = (F, \mu)$, we define $\text{reach}(\chi)$ as the set of all perfect matchings μ' of F such that $\mu \rightsquigarrow_F \mu'$. For any configuration $\chi = (F, \mu)$ and any matching μ' of F, we define $\text{reach}(\chi, \mu')$ as the set of all matchings μ'' in $\text{reach}(\chi)$ such that μ'' contains μ'.

We now state the three reachability problems studied in this chapter.

- The reachable matching problem: Given a configuration $\chi = (F, \mu)$ and a perfect matching μ' of F, determine whether μ' belongs to reach(χ).

- The reachable object problem: Given a configuration $\chi = (F, \mu)$ where $F = (A, B, \succ, E)$, an agent a in A, and an object b in B, determine whether there is a matching μ' in reach(χ) such that $\mu'(a) = b$.

- The Pareto-efficient matching problem: Given a configuration χ, find a matching in reach(χ) that is not Pareto-dominated by any other matching in reach(χ).

2.3 Reachability Over a Path Network

We begin by introducing some notation.

For any non-negative integer n, we define $[n]$ as $\{1, \ldots, n\}$. Without loss of generality, in this section we restrict attention to OAFs of the form $([n], [n], \succ, \{(b, b+1) \mid 1 \leq b < n\})$ for some positive integer n. We use the notation $(n, \succ)$ to refer to such an OAF.

For any non-negative integer n, we define $\Phi(n)$ as the set of all matchings μ such that agents(μ) = $[|\mu|]$ and objects(μ) $\subseteq [n]$.

For any matching μ in $\Phi(n)$, we define max(μ) as the maximum matched object in objects(μ), or as 0 if $\mu = \emptyset$.

For any matching μ in $\Phi(n)$, we define hole(μ) as the minimum positive integer that does not belong to objects(μ).

For any matching μ in $\Phi(n)$ and any agent a in agents(μ), define span(μ, a) as $\{b \in [n] \mid \mu(a) \leq b \leq a\} \cup \{b \in [n] \mid a \leq b \leq \mu(a)\}$.

For any OAF $F = (n, \succ)$, we define μ_F as the matching $\{(i, i) \mid i \in [n]\}$, and we define χ_F as the configuration (F, μ_F).

For any OAF $F = (n, \succ)$ and any agent a in $[n]$, we define left($\succ, a$) as the minimum object b in $[n]$ such that $b \succ_a b+1 \succ_a \cdots \succ_a \mu_F(a) = a$, and we define right($\succ, a$) as the maximum object b in $[n]$ such that $b \succ_a b-1 \succ_a \cdots \succ_a a$. Thus if a matching μ belongs to reach(χ_F), then the match $\mu(a)$ of agent a is at least left($\succ, a$) and at most right($\succ, a$), regardless of the preferences of the remaining agents.

For any OAF $F = (n, \succ)$, any matching μ in $\Phi(n)$, and any agent a in agents(μ), we say that the predicate IR($\succ, \mu, a$) holds (where "IR" stands for "individually rational") if left($\succ, a$) $\leq \mu(a) \leq$ right($\succ, a$). We say that the predicate IR($\succ, \mu$) holds if IR($\succ, \mu, a$) holds for all agents a in agents(μ).

2.3.1 A Useful Subroutine

This section presents Algorithm 1, a greedy subroutine that we use in Sections 2.3.3 and 2.3.4 to solve reachability problems over a path network.

Recall that the reachable object problem with path configuration χ_F is to check whether an object b is reachable for an agent a. Algorithm 1 addresses a variant of this problem in which the agents less than a are all required to be matched to specific objects. The input matching μ_0 specifies the required match for each of these agents.

Algorithm 1: A greedy path reachability subroutine.

Input: An OAF $F = (n, \succ)$, a matching μ_0 in $\Phi(n)$ such that $|\mu_0| < n$ and $\text{reach}(\chi_F, \mu_0) \neq \emptyset$, and a matching $\mu_1 = \mu_0 + (|\mu_1|, b_0)$ where $\max(\mu_0) < b_0 \leq \text{right}(\succ, |\mu_1|)$

Output: A matching μ in $\text{reach}(\chi_F, \mu_1)$, or $\emptyset$ if this set is empty

$\mu = \mu_1$;
while $0 < |\mu| < n$ **do**
  if $\text{left}(\succ, |\mu| + 1) \leq \text{hole}(\mu)$ **then**
    $\mu = \mu + (|\mu| + 1, \text{hole}(\mu))$;
  else if $\max(\mu) < \text{right}(\succ, |\mu| + 1)$ **then**
    $\mu = \mu + (|\mu| + 1, \max(\mu) + 1)$;
  else
    $\mu = \emptyset$;
  end
end
return μ

2.3.2 Proof of Correctness of Algorithm 1

In this section, we establish the correctness of Algorithm 1.

We begin by defining a specific subset $\Phi^*(n)$ of $\Phi(n)$. For any matching μ in $\Phi(n)$ and any integer i in $[|\mu|]$, let μ_i be the matching such that $\mu_i \subseteq \mu$ and $\text{agents}(\mu) = [i]$. Then $\Phi^*(n)$ is the set of all matchings μ such that μ belongs to $\Phi(n)$ and for each i in $[|\mu| - 1]$, either $\mu(i + 1) = \text{hole}(\mu_i)$ or $\max(\mu_i) < \mu(i + 1) \leq n$.

We now present a number of useful structural properties of matchings in $\Phi^*(n)$.

Representing a Matching as a Pair of Binary Strings

For any binary string α, we let $|\alpha|$ denote the length of α, and we let $w(\alpha)$ denote the Hamming weight of α. For any binary string α and any integer i in $[|\alpha|]$, we let α_i denote bit i of α. For any binary string α, and any integers i and j in $[|\alpha|]$, we let $\alpha_{i,j}$ denote the substring $\alpha_i \cdots \alpha_j$ of α.

For any integers m and n such that $0 \leq m \leq n$, we let $\Psi(m,n)$ denote the set of all pairs of binary strings (α, β) such that $|\alpha| = m$, $|\beta| = n$, $w(\alpha_{1,i}) \geq w(\beta_{1,i})$ holds for all i in $[m]$, $w(\alpha) = w(\beta)$, and $m < n$ implies $\beta_n = 1$.

For any (α, β) in $\Psi(m,n)$, we define $[\alpha, \beta]$ as the cardinality-m matching μ in $\Phi(n)$ constructed as follows: for any agent a in $[m]$ such that α_a is the ith 0 (resp., 1) in α, we define $\mu(a)$ as the index of the ith 0 (resp., 1) in β.

Observation 2.3.1. *Let (α, β) belong to $\Psi(m,n)$, let μ denote $[\alpha, \beta]$, and let b belong to $[n]$. If b is unmatched in μ, then $\beta_b = 0$. Otherwise, the following conditions hold, where a denotes $\mu^{-1}(b)$: $a > b$ implies $\alpha_a = \beta_b = 0$, $a < b$ implies $\alpha_a = \beta_b = 1$, and $a = b$ implies $\alpha_a = \beta_a$.*

Observation 2.3.2. *Let (α, β) belong to $\Psi(m,n)$ and let μ denote $[\alpha, \beta]$. If $m < n$ then $(\alpha 0, \beta)$ belongs to $\Psi(m+1, n)$ and $[\alpha 0, \beta] = \mu + (m+1, \text{hole}(\mu))$. Furthermore, for any non-negative integer k, $(\alpha 1, \beta 0^k 1)$ belongs to $\Psi(m+1, n+k+1)$ and $[\alpha 1, \beta 0^k 1] = \mu + (m+1, n+k+1)$.*

For any (α, β) in $\Psi(m,n)$, and any agent a in $[m]$ such that $\alpha_a = \beta_a$ and $w(\alpha_{1,a}) = w(\beta_{1,a})$, we say that a complement operation is applicable to

(α, β) at agent a. The result of applying this operation is the pair of binary strings (α', β') that is the same as (α, β) except $\alpha'_a = \beta'_a = 1 - \alpha_a$. It is easy to see that (α', β') belongs to $\Psi(m, n)$.

For any (α, β) and (α', β') in $\Psi(m, n)$, we write $(\alpha, \beta) \simeq (\alpha', \beta')$ to denote that (α, β) can be transformed into (α', β') via a sequence of complement operations.

Observation 2.3.3. *Let (α, β) and (α', β') belong to $\Psi(m, n)$. Then $[\alpha, \beta] = [\alpha', \beta']$ if and only if $(\alpha, \beta) \simeq (\alpha', \beta')$.*

Observation 2.3.4. *Let (α, β) belong to $\Psi(m, n)$, and let (α', β') belong to $\Psi(m', n')$ where $m \leq m'$ and $n \leq n'$. Then $[\alpha, \beta] \subseteq [\alpha', \beta']$ if and only if $(\alpha, \beta) \simeq (\alpha'_{1,|\alpha|}, \beta'_{1,|\beta|})$.*

For any (α, β) in $\Psi(m, n)$, and any object b in $[n-1]$ such that $\beta_b = 1$ and $\beta_{b+1} = 0$, we say that a sort operation is applicable to (α, β) across objects b and $b+1$. The result of applying this operation is the pair of binary strings (α, β') that is the same as (α, β) except $\beta'_b = 0$ and $\beta'_{b+1} = 1$.

Observation 2.3.5. *Let (α, β) belong to $\Psi(m, n)$ and let (α, β') be the result of applying a sort operation to (α, β) across objects b and $b+1$. Then (α, β') belongs to $\Psi(m, n)$. Furthermore, if $m = n$ then*

$$[\alpha, \beta'] = (\mu \setminus \{(a, b), (a', b+1)\}) \cup \{(a', b), (a, b+1)\}$$

where μ denotes $[\alpha, \beta]$, a denotes $\mu^{-1}(b)$, and a' denotes $\mu^{-1}(b+1)$.

For any (α, β) and (α', β') in $\Psi(m, n)$, we write $(\alpha, \beta) \rightsquigarrow (\alpha', \beta')$ to denote that (α, β) can be transformed into (α', β') via a sequence of complement and sort operations.

Observation 2.3.6. *Let α be a binary string of length m, and let β be a binary string of length n such that $m \leq n$. Then (α, β) belongs to $\Psi(m, n)$ if and only if $(0^m, 0^n) \rightsquigarrow (\alpha, \beta)$.*

For any non-negative integer n, we define $\Phi'(n)$ as the set of all matchings μ in $\Phi(n)$ such that $\mu = [\alpha, \beta]$ for some (α, β) in $\Psi(|\mu|, \max(\mu))$.

For any (α, β) in $\Psi(n, n)$, and any agent a in $[n-1]$ such that $w(\alpha_{1,a}) > w(\beta_{1,a})$ and $\alpha_a = 1$, we say that a pivot operation is applicable to (α, β) at agent a. The result of applying this operation is the pair of binary strings (α', β) that is the same as (α, β) except $\alpha'_a = 0$ and $\alpha'_{a'} = 1$, where a' denotes the minimum agent greater than a for which $w(\alpha_{1,a'}) = w(\beta_{1,a'})$. (The agent a' is well-defined since $w(\alpha) = w(\beta)$.)

Observation 2.3.7. *Let (α, β) belong to $\Psi(n, n)$, and let (α', β) be the result of applying a pivot operation to (α, β) at agent a. Then (α', β) belongs to $\Psi(n, n)$ and $\mathrm{span}([\alpha', \beta], a')$ is contained in $\mathrm{span}([\alpha, \beta], a')$ for all agents a' in $[n] - a$.*

For any (α, β) in $\Psi(n, n)$, and any object b in $[n-1]$ such that $w(\alpha_{1,b}) > w(\beta_{1,b})$, $\beta_b = 0$, and $\beta_{b+1} = 1$, we say that an unsort operation is applicable to (α, β) across objects b and $b + 1$. The result of applying this operation is

the pair of binary strings (α, β') that is the same as (α, β) except $\beta'_b = 1$ and $\beta'_{b+1} = 0$.

Observation 2.3.8. *Let (α, β) belong to $\Psi(n, n)$, and let (α, β') be the result of applying an unsort operation to (α, β) across objects b and $b + 1$. Then (α, β') belongs to $\Psi(n, n)$ and $\text{span}([\alpha, \beta'], a)$ is contained in $\text{span}([\alpha, \beta], a)$ for all agents a in $[n]$.*

Structural Properties of Matchings in $\Phi^*(n)$

Claim 2.3.9 below gives a a simple characterization of matchings in $\Phi'(n)$, and hence implies that $\Phi'(n) = \Phi^*(n)$.

Claim 2.3.9. *Let μ be a matching in $\Phi'(n)$ such that $|\mu| < n$, let a denote $|\mu|$, let a' denote $a + 1$, let b denote $\max(\mu)$, and let μ' denote $\mu + (a', b^*)$. Then μ' belongs to $\Phi'(n)$ if and only if $b^* = \text{hole}(\mu)$ or $b < b^* \leq n$.*

Proof. Since μ belongs to $\Phi'(n)$, there exists (α, β) in $\Psi(a, b)$ such that $\mu = [\alpha, \beta]$. Let b' denote $\max(\mu')$.

For the "if" direction, we need to prove that there exists (α', β') in $\Psi(a', b')$ such that $\mu' = [\alpha', \beta']$. We consider two cases.

Case 1: $b^* = \text{hole}(\mu) \leq b$. Observation 2.3.2 implies that $(\alpha 0, \beta)$ belongs to $\Psi(a', b')$ and $\mu' = [\alpha 0, \beta]$.

Case 2: $b < b^* \leq n$. Let k denote $b^* - b - 1$. Observation 2.3.2 implies that $(\alpha 1, \beta 0^k 1)$ belongs to $\Psi(a', b')$ and $\mu' = [\alpha 1, \beta 0^k 1]$.

We now address the "only if" direction. Assume that μ' belongs to $\Phi'(n)$. Since $\mu' = \mu + (a', b^*)$, we deduce that b^* belongs to $[n] \setminus \text{objects}(\mu)$. It remains to prove that $b^* = \text{hole}(\mu)$ or $b < b^*$. Let B denote the set of objects $[b] \setminus \text{objects}(\mu)$. We consider two cases.

Case 1: $a = b$. Thus $B = \emptyset$ and since b^* is unmatched in μ, we have $b < b^*$.

Case 2: $a < b$. Since μ' belongs to $\Phi'(n)$, there exists (α', β') in $\Psi(a', b')$ such that $\mu' = [\alpha', \beta']$. Observation 2.3.4 implies that $(\alpha'_{1,a}, \beta'_{1,b})$ belongs to $\Psi(a, b)$ and $(\alpha'_{1,a}, \beta'_{1,b}) \simeq (\alpha, \beta)$. Since $(\alpha'_{1,a}, \beta'_{1,b}) \simeq (\alpha, \beta)$, Observation 2.3.3 implies that $[\alpha'_{1,a}, \beta'_{1,b}] = [\alpha, \beta] = \mu$. Let B_0 denote the set of all objects i in B such that $\beta'_i = 0$. Observation 2.3.1 implies that $B_0 = B$. We consider two cases.

Case 2.1: $\alpha'_{a'} = 1$. Since $\mu'(a') = b^*$, Observation 2.3.1 implies that $\beta'_{b^*} = 1$. Thus b^* does not belong to $B = B_0$. Since b^* is unmatched in μ, we conclude that $b < b^*$.

Case 2.2: $\alpha'_{a'} = 0$. Since $a < b$ and $(\alpha'_{1,a}, \beta'_{1,b})$ belongs to $\Psi(a, b)$, we have $\beta'_b = 1$ and $w(\alpha'_{1,a}) = w(\beta'_{1,b}) > w(\beta'_{1,a})$. Let k denote the number of 0's in $\alpha'_{1,a}$, and let ℓ denote the number of 0's in $\beta'_{1,a}$. Since $w(\alpha'_{1,a}) > w(\beta'_{1,a})$, we have $\ell > k$. Let B' denote the indices of the first k 0's in $\beta'_{1,a}$, and let B'' denote the indices of the remaining $\ell - k$ 0's in $\beta'_{1,a}$. Since $\mu = [\alpha'_{1,a}, \beta'_{1,b}]$, we deduce that the objects in B' are all matched in μ and the objects in B'' are all unmatched in μ. Thus $B \cap [a] = B_0 \cap [a] = B'' \neq \emptyset$, It follows that hole$(\mu)$ is

the minimum object in B''. Since $\mu' = [\alpha', \beta']$, $\alpha'_{a'} = 0$, and there are $k+1$ 0's in α', we deduce that b^* is the minimum object in B''. Thus $b^* = \text{hole}(\mu)$. $\square$

Lemma 2.3.10 below establishes a one-to-one correspondence between matchings that are reachable from χ_F and matchings in $\Phi^*(n)$.

Lemma 2.3.10. *Let $F = (n, \succ)$ be an OAF. Then μ belongs to* $\text{reach}(\chi_F)$ *if and only if μ belongs to $\Phi^*(n)$, $|\mu| = n$, and* $\text{IR}(\succ, \mu)$ *holds.*

Proof. First we prove the "only if" direction. Suppose that μ belongs to $\text{reach}(\chi_F)$. Then there is a sequence $\mu_F = \mu_0, \ldots, \mu_k = \mu$ of perfect matchings of F such that $\mu_{i-1} \rightarrow_F \mu_i$ for all i in $[k]$.

For any i such that $0 \leq i \leq k$, let $P(i)$ denote the predicate asserting that the following conditions hold: μ_i belongs to $\Phi^*(n)$; $|\mu_i| = n$; $\text{IR}(\succ, \mu_i)$ holds. We prove by induction on i that $P(i)$ holds for all i in $\{0, \ldots, k\}$. Using the definition of μ_F, it is easy to see that $P(0)$ holds. Now consider the induction step. Fix i in $[k]$ and assume that $P(i-1)$ holds. We need to prove that $P(i)$ holds. Since $P(i-1)$ holds, we know that μ_{i-1} belongs to $\Phi^*(n)$, $|\mu_{i-1}| = n$, and $\text{IR}(\succ, \mu_{i-1})$. Since μ_{i-1} belongs to $\Phi^*(n)$ and $|\mu_{i-1}| = n$, there exists (α, β) in $\Psi(n, n)$ such that $\mu_{i-1} = [\alpha, \beta]$. Let b denote the object in $[n-1]$ such that $\mu_{i-1} \rightarrow_{F,(b,b+1)} \mu_i$.

Let (α', β') and (α'', β'') be defined as follows. First, if a complement operation is applicable to (α, β) at agent b and $\beta_b = 0$, then (α', β') is the result of applying this operation to (α, β), and otherwise (α', β') is equal to (α, β).

Second, if a complement operation is applicable to (α, β) at agent $b+1$ and $\beta_{b+1} = 1$, then (α'', β'') is the result of applying this operation to (α', β'), and otherwise (α'', β'') is equal to (α', β'). Observation 2.3.3 implies that (α'', β'') belongs to $\Psi(n, n)$ and $\mu_{i-1} = [\alpha'', \beta'']$.

Using Observation 2.3.1, it is straightforward to prove that $\beta''_b = 1$ and $\beta''_{b+1} = 0$. Thus a sort operation is applicable to (α'', β'') across objects b and $b+1$; let (α'', β''') denote the result of applying this sort operation. Observation 2.3.5 implies that (α'', β''') belongs to $\Psi(n, n)$ and $\mu_i = [\alpha'', \beta''']$. Thus μ_i belongs to $\Phi^*(n)$ and $|\mu_i| = n$. Since $\mathrm{IR}(\succ, \mu_{i-1})$ holds and $\mu_{i-1} \rightarrow_F \mu_i$, we deduce that $\mathrm{IR}(\succ, \mu_i)$ holds. We conclude that $P(i)$ holds, completing the proof by induction.

We now prove the "if" direction. Assume that μ belongs to $\Phi^*(n)$, $|\mu| = n$, and $\mathrm{IR}(\succ, \mu)$ holds. Observation 2.3.6 implies there exists (α, β) in $\Psi(n, n)$ such that $(0^n, 0^n) \rightsquigarrow (\alpha, \beta)$. It follows that there is a sequence $(0^n, 0^n) = (\alpha^{(0)}, \beta^{(0)}), \ldots, (\alpha^{(k)}, \beta^{(k)}) = (\alpha, \beta)$ of pairs in $\Psi(n, n)$ such that an applicable complement or sort operation transforms $(\alpha^{(i-1)}, \beta^{(i-1)})$ into $(\alpha^{(i)}, \beta^{(i)})$ for all i in $[k]$. Let μ_i denote $[\alpha^{(i)}, \beta^{(i)}]$ for all i such that $0 \leq i \leq k$. Thus $\mu_0 = \mu_F$ and $\mu_k = \mu$.

Observations 2.3.3 and 2.3.5 imply that for all i in $[k]$, either $\mu_i = \mu_{i-1}$ or μ_i is obtained from μ_{i-1} via an exchange across two adjacent objects. It remains to prove that any such exchanges are swaps, i.e., do not violate individual rationality. Below we accomplish this by proving that $\mathrm{IR}(\succ, \mu_i)$ holds for $0 \leq i \leq k$.

For any i in $[k]$, let $P(i)$ denote the predicate $\text{span}(\mu_{i-1}, a) \subseteq \text{span}(\mu_i, a)$ for all agents a in $[n]$. We claim that $P(i)$ holds for all i in $[k]$. To prove the claim, fix an integer i in $[k]$. We consider two cases.

Case 1: A complement operation transforms $(\alpha^{(i-1)}, \beta^{(i-1)})$ into $(\alpha^{(i)}, \beta^{(i)})$. In this case, Observation 2.3.3 implies that $\mu_i = \mu_{i-1}$. Thus $\text{span}(\mu_{i-1}, a) = \text{span}(\mu_i, a)$ for all agents a in $[n]$.

Case 2: A sort operation transforms $(\alpha^{(i-1)}, \beta^{(i-1)})$ into $(\alpha^{(i)}, \beta^{(i)})$. Assume that the sort operation is applied to $(\alpha^{(i-1)}, \beta^{(i-1)})$ across objects b and $b+1$. Then $\beta_b^{(i-1)} = 1$ and $\beta_{b+1}^{(i)} = 0$, and Observation 2.3.1 implies that $\mu_{i-1}^{-1}(b) \leq b$ and $\mu_{i-1}^{-1}(b+1) \geq b+1$. Thus Observation 2.3.5 implies $\text{span}(\mu_{i-1}, a) \subseteq \text{span}(\mu_i, a)$ for all agents a in $[n]$.

Since $P(i)$ holds for all i in $[k]$ and $\text{IR}(\succ, \mu_k)$ holds, we deduce that $\text{IR}(\succ, \mu_i)$ holds for all i such that $0 \leq i \leq k$, as required. □

The next three lemmas are concerned with enlarging a given matching μ in $\Phi^*(n)$ such that $|\mu| < n$ by introducing a suitable match for agent $|\mu|+1$. Lemma 2.3.11 (resp., Lemma 2.3.12) addresses the case where agent $|\mu|+1$ is matched to an object that is at most (resp., at least) $|\mu|+1$. By combining these two lemmas, we obtain Lemma 2.3.13, which shows that it is sufficient to consider matching agent $|\mu|+1$ with an object in $\{\text{hole}(\mu), \max(\mu)+1\}$.

Lemma 2.3.11. *Let $F = (n, \succ)$ be an OAF, let μ be a matching in $\Phi^*(n)$ such that $|\mu| < n$, let a denote $|\mu|$, let a' denote $a+1$, let b denote* $\max(\mu)$,

and let μ' *denote* $\mu + (a', \text{hole}(\mu))$. *Assume that* $a < b$, $\text{reach}(\chi_F, \mu) \neq \emptyset$, *and* $\text{left}(\succ, a') \leq \text{hole}(\mu)$. *Then* $\text{reach}(\chi_F, \mu') \neq \emptyset$.

Proof. Let μ^* be a matching in $\text{reach}(\chi_F, \mu)$. Lemma 2.3.10 implies that $\text{IR}(\succ, \mu^*)$ holds and there exists (α^*, β^*) in $\Psi(n, n)$ such that $\mu^* = [\alpha^*, \beta^*]$. Using Observations 2.3.3 and 2.3.4, we deduce that $[\alpha^*_{1,a}, \beta^*_{1,b}]$ is equal to μ. We consider two cases.

Case 1: $\alpha^*_{a'} = 0$. Using Observation 2.3.2, we deduce that $[\alpha^*_{1,a'}, \beta^*_{1,b}]$ is equal to μ'. Hence Observation 2.3.4 implies that μ^* contains μ'. Thus $\text{reach}(\chi_F, \mu') \neq \emptyset$, as required.

Case 2: $\alpha^*_{a'} = 1$. In this case, it is straightforward to prove that a pivot operation is applicable to (α^*, β^*) at agent a'; let (α^{**}, β^*) denote the result of this operation. Observation 2.3.7 implies that (α^{**}, β^*) belongs to $\Psi(n, n)$. Let μ^{**} denote $[\alpha^{**}, \beta^*]$. Thus μ^{**} belongs to $\Phi^*(n)$. Observation 2.3.7 implies that $\text{IR}(\succ, \mu^{**}, a'')$ holds for all agents a'' in $[n] - a'$. Since the inequality $\text{left}(\succ, a') \leq \text{hole}(\mu)$ implies that $\text{IR}(\succ, \mu^{**}, a')$ holds, we deduce that $\text{IR}(\succ, \mu^{**})$ holds. Thus Lemma 2.3.10 implies that μ^{**} belongs to $\text{reach}(\chi_F)$. Since $\alpha^{**}_{1,a} = \alpha^*_{1,a}$, Observation 2.3.4 implies that μ^{**} contains μ. Using Observation 2.3.2, we deduce that $[\alpha^{**}_{1,a'}, \beta^*_{1,b}]$ is equal to μ'. Hence Observation 2.3.4 implies that μ^{**} contains μ'. Since μ^{**} belongs to $\text{reach}(\chi_F)$ and μ^{**} contains μ', we conclude that μ^{**} is contained in $\text{reach}(\chi_F, \mu')$. □

Lemma 2.3.12. *Let* $F = (n, \succ)$ *be an OAF, let* μ *be a matching in* $\Phi^*(n)$ *such that* $|\mu| < n$, *let* a *denote* $|\mu|$, *let* a' *denote* $a + 1$, *let* b *denote* $\max(\mu)$,

let μ' *denote* $\mu + (a', b')$ *where* b' *belongs to* $[n] \setminus [b+1]$ *and* $\text{reach}(\chi_F, \mu') \neq \emptyset$*, and let* μ'' *denote* $\mu + (a', b+1)$*. Then* $\text{reach}(\chi_F, \mu'') \neq \emptyset$*.*

Proof. Let μ^* be a matching in $\text{reach}(\chi_F, \mu')$. Lemma 2.3.10 implies that $\text{IR}(\succ, \mu^*)$ holds and there exists (α^*, β^*) in $\Psi(n, n)$ such that $\mu^* = [\alpha^*, \beta^*]$. Using Observations 2.3.1 and 2.3.4, we find that $\alpha^*_{a'} = \beta^*_{b'} = 1$ and $\beta^*_{b+k} = 0$ for all k in $[b' - b - 1]$. Using Observation 2.3.8, we deduce that a sequence of $b' - b - 1$ unsort operations can be used to transform (α^*, β^*) into a pair of binary strings (α^*, β^{**}) in $\Psi(n, n)$ such that $\beta^{**}_{1,b} = \beta^*_{1,b}$ and $\text{IR}(\succ, [\alpha^*, \beta^{**}])$ holds. Let μ^{**} denote $[\alpha^*, \beta^{**}]$; thus μ^{**} belongs to $\Phi^*(n)$. Lemma 2.3.10 implies that μ^{**} belongs to $\text{reach}(\chi_F)$. Since $\beta^{**}_{1,b} = \beta^*_{1,b}$, Observation 2.3.4 implies that μ^{**} contains μ. Using Observation 2.3.2, we deduce that $[\alpha^*_{1,a'}, \beta^{**}_{1,b+1}]$ is equal to μ''. Hence Observation 2.3.4 implies that μ^{**} contains μ''. Since μ^{**} belongs to $\text{reach}(\chi_F)$ and μ^{**} contains μ'', we conclude that μ^{**} belongs to $\text{reach}(\chi_F, \mu'')$. □

Lemma 2.3.13. *Let* $F = (n, \succ)$ *be an OAF, let* μ *be a matching in* $\Phi^*(n)$ *such that* $|\mu| < n$*, let* a *denote* $|\mu|$*, let* a' *denote* $a + 1$*, let* b *denote* $\max(\mu)$*, and assume that* $\text{reach}(\chi_F, \mu) \neq \emptyset$*. Then there exists a matching in* $\text{reach}(\chi_F, \mu)$ *that matches agent* a' *to an object in* $\{\text{hole}(\mu), b+1\}$*.*

Proof. Let μ^* belong to $\text{reach}(\chi_F, \mu)$ and let b^* denote $\mu^*(a')$. Lemma 2.3.10 implies that μ^* belongs to $\Phi^*(n)$ and $\text{IR}(\succ, \mu^*)$ holds.

Since μ^* belongs to $\Phi^*(n)$, the definition of $\Phi^*(n)$ implies that $b^* = \text{hole}(\mu)$ or $b < b^*$. We consider two cases.

Case 1: $b^* = \text{hole}(\mu)$. Since $\text{IR}(\succ, \mu^*)$ holds, we deduce that $\text{left}(\succ, a') \leq b^*$. Hence the claim of the lemma follows from Lemma 2.3.11.

Case 2: $b < b^*$. Since $\text{IR}(\succ, \mu^*)$ holds, we deduce that $b^* \leq \text{right}(\succ, a')$. If $b^* = b + 1$, the claim of the lemma is immediate. Otherwise, it follows from Lemma 2.3.12. □

We are now ready to prove the main technical lemma of this section, Lemma 2.3.14 below.

Lemma 2.3.14. *Consider an execution of Algorithm 1 with inputs $F = (n, \succ)$, μ_0, and μ_1. If the guard of the while loop is evaluated in a state where $\mu \neq \emptyset$, then the following conditions hold in that state: (1) μ belongs to $\Phi^*(n)$; (2) $\text{IR}(\succ, \mu)$ holds; (3) $\text{reach}(\chi_F, \mu_1) \neq \emptyset$ implies $\text{reach}(\chi_F, \mu) \neq \emptyset$. Furthermore, if the guard of the while loop is evaluated in a state where $\mu = \emptyset$, then $\text{reach}(\chi_F, \mu_1) = \emptyset$.*

Proof. We prove the claim by induction on the number of iterations of the loop. For the base case, we verify that the stated conditions hold the first time the loop is reached. The initialization of μ ensures that $\mu \neq \emptyset$, so we need to verify conditions (1) through (3). Since $\text{reach}(\chi_F, \mu_0) \neq \emptyset$, Lemma 2.3.10 implies that $\text{IR}(\succ, \mu_0)$ holds, and Lemma 2.3.10 and the definition of $\Phi^*(n)$ together imply that μ_0 belongs to $\Phi^*(n)$. Since $b_0 \leq \text{right}(\succ, |\mu_1|) \leq n$, the definition of $\Phi^*(n)$ implies that μ_1 belongs to $\Phi^*(n)$. Since $\text{IR}(\succ, \mu_0)$ holds and the preconditions associated with Algorithm 1 ensure that $\text{IR}(\succ, \mu_1, |\mu_1|)$

holds, we deduce that $\mathrm{IR}(\succ, \mu_1)$ holds. Since μ is initialized to μ_1, we conclude that conditions (1) through (3) hold the first time the loop is reached.

For the induction step, consider an arbitrary iteration of the loop. Such an iteration begins in a state where the guard of the while loop evaluates to true, so we can assume that $0 < |\mu| < n$ and that conditions (1) through (3) hold in this state. Let a denote $|\mu|$, let a' denote $a + 1$, let b denote $\max(\mu)$, and let μ' denote the value of the program variable μ immediately after this iteration of the loop body. We consider two cases.

Case 1: $\mu' = \emptyset$. In this case, we need to prove that $\mathrm{reach}(\chi_F, \mu_1) = \emptyset$. Assume for the sake of contradiction that $\mathrm{reach}(\chi_F, \mu_1) \neq \emptyset$. Condition (3) implies that $\mathrm{reach}(\chi_F, \mu) \neq \emptyset$. Thus Lemma 2.3.13 implies there exists a matching μ^* in $\mathrm{reach}(\chi_F, \mu)$ such that $\mathrm{left}(\succ, a') \leq \mathrm{hole}(\mu)$ or $b + 1 \leq \mathrm{right}(\succ, a')$. It follows by inspection of the code that $\mu' \neq \emptyset$, a contradiction.

Case 2: $\mu' \neq \emptyset$. In this case, we need to prove that conditions (1) through (3) hold with μ replaced by μ'; we refer to these conditions as postconditions (1) through (3). Since $\mu'(a') \leq \mathrm{right}(\succ, a') \leq n$ and condition (1) implies that μ belongs to $\Phi^*(n)$, the definition of $\Phi^*(n)$implies that μ' belongs to $\Phi^*(n)$. Thus postcondition (1) holds. Since condition (2) implies that $\mathrm{IR}(\succ, \mu)$ holds, and $\mathrm{IR}(\succ, \mu', a')$ holds by inspection of the code, we deduce that postcondition (2) holds. It remains to establish postcondition (3). In order to do so, we may assume that $\mathrm{reach}(\chi_F, \mu_1) \neq \emptyset$. Condition (3) implies that $\mathrm{reach}(\chi_F, \mu) \neq \emptyset$. To establish postcondition (3), we need to prove that $\mathrm{reach}(\chi_F, \mu') \neq \emptyset$. We consider two cases.

Case 2.1: $a < b$ and $\text{left}(\succ, a') \leq \text{hole}(\mu)$. In this case, $\mu' = \mu + (a', \text{hole}(\mu))$, and Lemma 2.3.11 implies that $\text{reach}(\chi_F, \mu') \neq \emptyset$.

Case 2.2: $a = b$ or $\text{hole}(\mu) < \text{left}(\succ, a')$. In this case, $\mu' = \mu + (a', b+1)$, and Lemma 2.3.13 implies there exists a matching μ^* in $\text{reach}(\chi_F, \mu)$ such that $\mu^*(a')$ is either $\text{hole}(\mu)$ or $b+1$. Since $\text{IR}(\succ, \mu^*)$ holds, the Case 2.2 condition implies that if $\mu^*(a') = \text{hole}(\mu)$, then $a = b$, in which case $\text{objects}(\mu) = [b]$ and hence $\text{hole}(\mu) = b + 1$. It follows that $\mu^*(a') = b + 1$, and hence that $\text{reach}(\chi_F, \mu') \neq \emptyset$. □

Using Lemma 2.3.14, it is straightforward to establish the correctness of Algorithm 1.

Lemma 2.3.15. *Consider an execution of Algorithm 1 with inputs $F = (n, \succ)$, μ_0, and μ_1. The execution terminates correctly within $n - |\mu_1|$ iterations.*

Proof. Lemma 2.3.14 implies that each iteration of Algorithm 1 either increments the cardinality of matching μ or reduces it to zero. In the latter case, the algorithm terminates immediately. It follows that the algorithm terminates within $n - |\mu_1|$ iterations. Next, we argue that the algorithm terminates correctly. In what follows, let μ^* denote the final value of the program variable μ. We consider two cases.

Case 1: Algorithm 1 terminates with $\mu^* = \emptyset$. In this case, Lemma 2.3.14 implies that $\text{reach}(\chi_F, \mu_1) = \emptyset$, as required.

Case 2: Algorithm 1 terminates with $\mu^* \neq \emptyset$. Lemma 2.3.14 implies that conditions (1) through (3) in the statement of Lemma 2.3.14 hold with μ

replaced by μ^*; we refer to these conditions as postconditions (1) through (3). Since $\mu^* \neq \emptyset$, no agent-object pairs are removed from μ during the execution of Algorithm 1. Since Algorithm 1 initializes μ to μ_1, we deduce that μ_1 is contained in μ^*. Since the guard of the loop evaluates to false when μ is equal to μ^* and postcondition (1) holds, we deduce that μ^* belongs to $\Phi^*(n)$ and $|\mu^*| = n$. Since postconditions (1) and (2) hold, Lemma 2.3.10 implies that μ^* belongs to reach(χ_F). Since μ_1 is contained in μ^*, we deduce that μ^* belongs to reach(χ_F, μ_1), as required. □

2.3.3 Object Reachability

We now describe how to use Algorithm 1 to solve the reachable object problem on paths in $O(n^2)$ time. Let $F = (n, \succ)$ be a given OAF, let a^* be an agent in $[n]$, and let b^* be an object in $[n]$. Assume without loss of generality that $a^* < b^*$. We wish to determine whether there is a matching in reach(χ_F) that matches a^* to b^*, and if so, to compute such a matching. We begin by using a preprocessing phase to compute left$(\succ, a)$ and right$(\succ, a)$ for all agents a in $[n]$. We start with agent a^*, and check whether left$(\succ, a^*) \leq b^* \leq$ right$(\succ, a^*)$. If this check fails, we halt and report failure. Otherwise, we proceed to the remaining agents. Barring failure, the overall cost of the preprocessing phase is $O(n^2)$. We now describe how to proceed in the special case where a^* is the leftmost agent on the path, i.e., where $a^* = 1$. Later we will see how to efficiently reduce the general case to this special case. In the special case $a^* = 1$, we call Algorithm 1 with $\mu_0 = \emptyset$ and $\mu_1 = \{(1, b^*)\}$.

If there is a matching in reach(χ_F) that matches agent 1 to object b, then Algorithm 1 returns such a matching. If not, Algorithm 1 returns the empty matching. Excluding the cost of the preprocessing phase, the time complexity of Algorithm 1 is $O(n)$. So the overall running time is $O(n^2)$, as it is dominated by the preprocessing phase. We now discuss how to reduce the case of general a^* to the special case $a^* = 1$. The key is Lemma 2.3.16 below. Informally, Lemma 2.3.16 tells us that we can ignore all of the agents and objects in $[a^*-1]$. Doing this, a^* is once again leftmost, and we can proceed as in the special case $a^* = 1$. Alternatively, we can run Algorithm 1 with $\mu_0 = \{(i, i) \mid i \in [a^* - 1]\}$ and $\mu_1 = \mu_0 + \{(a^*, b^*)\}$. We now proceed to prove Lemma 2.3.16.

Lemma 2.3.16. *Let $F = (n, \succ)$ be an OAF, let μ be a matching in* reach(χ_F)*, let a belong to* agents(μ)*, let b denote $\mu(a)$, and assume that $a < b$. Then there is a matching μ' in* reach(χ_F) *such that $\mu'(a) = b$ and $\mu'(a') = a'$ for all agents a' in $[a - 1]$.*

For any (α, β) in $\Psi(n, n)$, we say that a cancel operation is applicable to (α, β) if $w(\alpha) > 0$. The result of applying this operation is the pair of binary strings (α', β') that is the same as (α, β) except that the first appearing 1 in α (resp., β) is changed to a 0 in α' (resp., β'). It is easy to see that (α', β') belongs to $\Psi(n, n)$.

Observation 2.3.17. *Let (α, β) belong to $\Psi(n, n)$, let a and a' be agents in $[n]$ such that $\alpha_a = \alpha_{a'} = 1$ and $a > a'$, let (α', β') be the result of applying a cancel operation to (α, β), let μ denote $[\alpha, \beta]$, and let μ' denote $[\alpha', \beta']$, Then*

$\mu'(a) = \mu(a)$ *and* $\text{span}(\mu', a'')$ *is contained in* $\text{span}(\mu, a'')$ *for all agents* a'' *in* $[n]$.

Proof. Lemma 2.3.10 implies that $\text{IR}(\succ, \mu)$ holds and there exists (α, β) in $\Psi(n, n)$ such that $\mu = [\alpha, \beta]$. Since $a < b$, Observation 2.3.1 implies that $\alpha_a = \beta_b = 1$. Let the number of 1 bits to the left of position a in α be equal to k. Thus the number of 1 bits to the left of position b in β is also k. Let (α', β') be the pair of binary strings in $\Psi(n, n)$ that results from applying k cancel operations to (α, β). Using Observation 2.3.17, it is straightforward to check that the matching $\mu' = [\alpha', \beta']$ satisfies the requirements of the lemma. □

Theorem 2.3.18 below summarizes the main result of this section.

Theorem 2.3.18. *Reachable object on paths can be solved in* $O(n^2)$ *time.*

2.3.4 Pareto-Efficient Reachability

Let $F = (n, \succ)$ be an OAF, and let $\chi = (F, \mu_0)$ be a configuration for which we wish to compute a Pareto-efficient matching. Below we describe a simple way to use Algorithm 1 to solve Pareto-efficient matching on paths in $O(n^3)$ time. We then explain how to improve the time bound to $O(n^2 \log n)$, and then to $O(n^2)$. In all cases we employ the same high-level strategy based on serial dictatorship. We begin by performing the $O(n^2)$-time preprocessing phase discussed in Section 2.3.3; we only need to perform this computation once. After the preprocessing phase, the output matching is computed in n stages numbered from 1 to n. In stage k, we determine the best possible match

that we can provide to agent k while continuing to maintain the previously-determined matches for agents 1 through $k-1$. We now describe how to use Algorithm 1 to implement any given stage k in $O(n^2)$ time. In stage k, we call Algorithm 1 $O(n)$ times. In each of these calls, the input matching μ_0 contains the $k-1$ previously-determined agent-object pairs involving the agents in $[k-1]$. The calls to Algorithm 1 differ only in terms of the value assigned to the input object b_0 which, together with μ_0, determines μ_1. We vary b_0 over all values meeting the precondition $\max(\mu_0) < b_0 \leq \text{right}(\succ, |\mu_1|)$ associated with Algorithm 1; the number of such values is $O(n)$.

This allows us to determine, in $O(n^2)$ time, the rightmost feasible match, if any, for agent k. By Lemma 2.3.13, the leftmost potential match for agent k is $\text{hole}(\mu_0)$, and this option is only feasible if $\text{left}(\succ, k) \leq \text{hole}(\mu_0)$. Since $\text{reach}(\chi_F, \mu_0) \neq \emptyset$, we are guaranteed to find at least one candidate match for agent k in this process. If there is exactly one candidate, then we select it as the match of agent k. Otherwise, there are two candidates (leftmost and rightmost), and we select the candidate that agent k prefers.

The simple algorithm described above has a running time of $O(n^2)$ per stage, and hence $O(n^3)$ overall. To understand how to implement a stage more efficiently, it is useful to assign a color to the program state each time the condition of the while loop is evaluated. We color such a state red if $\mu = \emptyset$. If the state is red, then the execution is guaranteed to fail (i.e., return $\emptyset$) immediately. We color such a state green if $|\mu| = \max(\mu) > 0$. If the state is green, it is straightforward to prove that the program will proceed

to assign every agent a in $\{|\mu|+1,\ldots,n\}$ to object a, and then will succeed (i.e., return a nonempty matching). This observation also implies that if the state is green after a given number of iterations, it remains green after each subsequent iteration. If a state is neither red nor green, we color it yellow.

We are now ready to see how to improve the running time of stage k to $O(n \log n)$. As a thought experiment, consider running the $O(n)$ executions of Algorithm 1 associated with the simple algorithm, but now in parallel. Within each of these executions, we color each successive agent in the set $\{k,\ldots,n\}$ white or black as follows: agent k is colored black; agent $|\mu|+1$ is colored white if $\text{left}(\succ, |\mu|+1) \leq \text{hole}(\mu)$, and black otherwise. The key observation is that as long as none of the parallel executions have terminated, they all agree on the coloring of the processed agents. It follows that if we compare two of the parallel executions, say executions A and B where execution A has a lower value for b_0 than execution B, then execution A can only transition to a green state at a strictly earlier iteration than execution B, and execution A cannot transition to a red state earlier than execution B. This implies that there is a threshold b_1 such that all executions with $b_0 \leq b_1$ succeed, and all of the remaining executions fail. This in turn means that we do not need to run all $O(n)$ of the parallel executions of Algorithm 1. Instead, we can use binary search to determine the threshold b_1 in $O(\log n)$ executions. This observation reduces the running time of a stage from $O(n^2)$ to $O(n \log n)$.

We now sketch how to further improve the running time of stage k to $O(n)$. To do so, we will use a single execution of a modified version of

Algorithm 1 to compute the threshold b_1 discussed above. The high-level idea is to treat b_0 as a variable instead of a fixed value. Initially, we set b_0 to $\max(\mu_0) + 1$, the minimum value satisfying the associated precondition of Algorithm 1. We also maintain a lower bound b_1' on the threshold b_1. We initialize b_1' to a low dummy value, such as 0. Each time the condition of the while loop is evaluated, we check whether the color of the current state is red, green, or yellow. If the color is yellow, we continue the execution without altering b_0 or b_1'. If the color is red, then we halt and output the threshold $b_1 = b_1'$. If the color is green, then we assign b_1' to the current value of b_0, and we increment b_0.

Unfortunately, we cannot simply increment b_0 and continue the execution. While incrementing b_0 has no impact on the white-black categorization of the agents processed so far, and on the matches of the white agents, it causes the match of each black agent to be incremented. This leads to two difficulties that we now discuss.

The first difficulty is that there can be a lot of black agents, making it expensive to maintain an explicit match for each black agent. Accordingly, when we color an agent black, we do not explicitly match that agent to a particular object. Instead, we maintain an ordered list of the black agents. At any given point in the execution, the black agents are implicitly matched (in the order specified by the list) to the contiguous block of objects that starts with the current value of b_0. Thus, when b_0 is incremented, the matches of the black agents are implicitly updated in constant time.

The second difficulty associated with incrementing b_0 is that if a black agent a is matched to object $\text{right}(\succ, a)$ just before the increment, then it is infeasible to shift the match of agent a to the right. If this happens, we need to recognize that executing Algorithm 1 from the beginning with the new higher value of b_0 results in a red state, and so we should terminate. To recognize such events, we introduce an integer variable called *slack*. We maintain the invariant that *slack* is equal to the maximum number of positions we can shift the list of black agents to the right without violating a constraint. We initialize *slack* to $\text{right}(\succ, |\mu_1|)$ minus the initial value $\max(\mu_0)+1$ of b_0. When we color an agent a black, we update *slack* to the minimum of its current value and $\text{right}(\succ, a) - b_0 - \ell$, where ℓ denotes the number of previously-identified black agents. When we increment b_0, we decrement *slack* in order to maintain the invariant. If *slack* becomes negative, we recognize that the program should be in a red state, and we terminate.

Upon termination, it is straightforward to argue that the output threshold b_1 is correct. If b_1 is equal to the initial dummy value, then the sole candidate match for agent k is object $\text{hole}(\mu_0)$. If not, object b_1 is a candidate, and if $\text{hole}(\mu_0)$ is not equal to b_1 and $\text{left}(\succ, k) \leq \text{hole}(\mu_0)$ then $\text{hole}(\mu_0)$ is a second candidate. If there are two candidates, we use the preferences of agent k to select between them.

Theorem 2.3.19 below summarizes the main result of this section.

Theorem 2.3.19. *Pareto-efficient matching on paths can be solved in $O(n^2)$ time.*

2.4 Pareto-Efficient Reachability on Generalized Stars

Throughout this section, let F denote an OAF associated with a generalized star G, let o denote the center object of G, let m denote the number of branches of G, and assume that the branches are indexed from 1 to m. For any i in $[m]$, let $\ell_i > 0$ denote the number of vertices on branch i. We refer to the objects on branch i as $\langle i, 1\rangle, \ldots, \langle i, \ell_i\rangle$, where object $\langle i, j\rangle$ is at distance j from the center. Let $\chi_0 = (F, \mu_0)$ denote the initial configuration, and let $n = 1 + \sum_{1 \le i \le m} \ell_i$ denote the total number of vertices in G.

Our algorithm uses serial dictatorship to compute a Pareto-efficient matching for configuration χ_0. For any sequence of agents $\sigma = a_1, \ldots, a_s$ we define serial(σ) as the cardinality-s matching of F in which agent a_1 (the first dictator) is matched to its best match b_1 in reach(χ_0, τ_0) where $\tau_0 = \emptyset$, agent a_2 (the second dictator) is matched to its best match b_2 in reach(χ_0, τ_1) where $\tau_1 = \tau_0 + (a_1, b_1)$, ..., and agent a_s (the sth dictator) is matched to its best match b_s in reach(χ_0, τ_{s-1}) where $\tau_{s-1} = \tau_{s-2} + (a_{s-1}, b_{s-1})$. Observe that for any permutation σ of the entire set of agents in F, serial(σ) is a Pareto-efficient matching of χ_0.

We iteratively grow a dictator sequence σ. We find it convenient to partition the iterations into two phases. The first phase ends when the current dictator is matched to the center object. The second phase reduces to solving a collection of disjoint path problems, one for each branch.

We begin by discussing the design and analysis of the first phase. We

find it useful to introduce the concept of a "nice pair" for OAF F. For any configuration χ of the form (F, μ) and any matching τ of F, we say that the pair (χ, τ) is nice for F if the following conditions hold:

- The set $\text{reach}(\chi, \tau)$ is nonempty.
- For any branch i in $[m]$, there is a (possibly empty) sequence of integers $1 \leq j_1 < \cdots < j_s \leq \ell_i$ such that $\tau(\chi(\langle i, t \rangle)) = \langle i, j_t \rangle$ for $1 \leq t \leq s$ and $\chi(\langle i, t \rangle)$ is unmatched in τ for $s < t \leq \ell_i$. We refer to this (unique) sequence as $\text{indices}(\chi, \tau, i)$.

The first phase iteratively updates a dictator sequence σ, a configuration χ, and a matching τ. We initialize the configuration χ to χ_0, the initial configuration of F. We initialize σ to the singleton sequence containing agent $\chi(o)$, the first dictator. We use Subroutine 1 of Section 2.4.1 to determine the best match of agent $\chi(o)$ in $\text{reach}(\chi)$, call it b, and we initialize τ to $\{(\chi(o), b)\}$.

We then execute the while loop described below, which we claim satisfies the following loop invariant I: (χ, τ) is a nice pair for F, agent $\chi(o)$ is matched in τ, $\text{reach}(\chi, \tau) = \text{reach}(\chi_0, \tau)$, and $\tau = \text{serial}(\sigma)$. It is straightforward to verify that invariant I holds after initialization of σ, χ, and τ. In Section 2.4.2, we prove that if I holds at the start of an iteration of the while loop, then I holds at the end of the iteration.

While $\tau(\chi(o)) \neq o$, we use the following steps to update σ, χ, and τ.

1. Since $\tau(\chi(o)) \neq o$, object $\tau(\chi(o))$ is of the form $\langle i, j_0 \rangle$ for some i in $[m]$ and j_0 in $[\ell_i]$. Let $j_1 < \cdots < j_s$ denote indices(χ, τ, i), and let a denote the agent $\chi(\langle i, s+1 \rangle)$.

2. Append agent a to the dictator sequence σ.

3. Use Subroutine 2 of Section 2.4.1 to set k to the maximum j such that object $\langle i, j \rangle$ is a possible match of agent a in reach(χ, τ), or to 0 if no such j exists.

4. If $\langle i, s+1 \rangle \prec_a \cdots \prec_a \langle i, 1 \rangle \prec_a o$, then perform the following steps.

   (a) Let μ denote the matching of F such that $\chi = (F, \mu)$, let μ^* denote the matching obtained from μ by applying $s+1$ swaps to move agent a from object $\langle i, s+1 \rangle$ to the center object o, and let χ^* denote the configuration (F, μ^*).

   (b) Use Subroutine 1 of Section 2.4.1 to set b to the best match of agent a in reach(χ^*, τ).

   (c) If $k = 0$ or $b \succ_a \langle i, k \rangle$, then set χ to χ^*, τ to $\tau + (a, b)$, and k to -1.

5. If $k > 0$, then set τ to $\tau + (a, \langle i, k \rangle)$.

Upon termination of the first phase, invariant I holds and $\tau(\chi(o)) = o$. Thus, letting χ_1 denote the value of program variable χ at the end of the first phase, we know that (χ_1, τ) is a nice pair for F, $\tau(\chi_1(o)) = o$, reach$(\chi_1, \tau) =$ reach(χ_0, τ), and $\tau = \text{serial}(\sigma)$.

In the second phase, we perform the following computation for each branch i (in arbitrary order). First, we let $j_1 < \cdots < j_s$ denote indices(χ_1, τ, i). Second, we perform the following steps for j ranging from $s+1$ to ℓ_i.

1. Let a denote agent $\chi_1(\langle i, j \rangle)$, and append a to σ.
2. Use Subroutine 3 of Section 2.4.1 to set b to the best match of agent a in reach(χ_1, τ).
3. Set τ to $\tau + (a, b)$.

At the end of the second phase, we output the matching τ.

Let I' denote the invariant "reach$(\chi_1, \tau) =$ reach(χ_0, τ) and $\tau =$ serial(σ)". Thus invariant I' holds at the end of the first phase. Moreover, it is easy to see that invariant I' continues to hold immediately after each execution of step 3 in the second phase.

Since invariant I holds in the first phase and invariant I' holds in the second phase, the overall algorithm faithfully implements the serial dictatorship framework discussed at the beginning of this section. Thus the algorithm correctly computes a Pareto-efficient matching for configuration χ_0. In Section 2.4.1, we explain how to implement Subroutines 1, 2, and 3 so that the overall running time of the algorithm is $O(n^2 \log n)$.

2.4.1 Polynomial-Time Implementation

In this section we describe an efficient implementation of the two-phase algorithm presented in Section 2.4. Our description of the first phase make use of Subroutines 1 and 2, while our description of the second phase makes use of Subroutine 3. Below we discuss how to implement Subroutines 1, 2, and 3 efficiently. Our analysis of the time complexity of these subroutines assumes that a certain preprocessing phase has been performed. Specifically, for each agent a in F, we precompute the set of all objects b such that the sequence of objects $\chi_0(a) = b_1, \ldots, b_k = b$ on the unique simple path from $\chi_0(a)$ (the initial object of agent a) to b in G satisfies $b_1 \prec_a \cdots \prec_a b_k$. It is straightforward to compute each such set in $O(n)$ time, and hence the overall time complexity of the preprocessing phase is $O(n^2)$.

We now describe Subroutine 1. The input to Subroutine 1 is a nice pair (χ, τ) for F such that agent $a = \chi(o)$ is unmatched in τ. The output of Subroutine 1 is the best match of a in reach(χ, τ). Subroutine 1 works by considering each branch i in turn to compute the best branch-i match of a in reach(χ, τ). For a fixed i in $[m]$, the latter problem can be solved as follows. Consider the path P of objects consisting of the center object o plus branch i. By restricting the generalized star configuration χ to path P, we obtain a path configuration χ_P. Similarly, by restricting the matching τ to the agents associated with path P, we obtain a matching τ_P defined on path P. For any given j in $[\ell_i]$, it is easy to argue that object $\langle i, j \rangle$ is a possible match of a in reach(χ, τ) if and only if object $\langle i, j \rangle$ is a possible match of a in

reach(χ_P, τ_P). Moreover, we can determine whether $\langle i, j\rangle$ is a possible match of a in reach(χ_P, τ_P) by performing at most one call to Algorithm 1 on path P. Given the results of the preprocessing phase, the additional time complexity required to determine whether $\langle i, j\rangle$ is a possible match of a in reach(χ_P, τ_P) is $O(\ell_i)$. Using binary search, we can determine the maximum j (if any) such that $\langle i, j\rangle$ is a possible match of a in reach(χ_P, τ_P) in $O(\ell_i \log \ell_i)$ time. (Remark: Letting $j_1 < \cdots < j_s$ denote indices(χ, τ, i), we can restrict the binary search to the interval $\{1, \ldots, j_1 - 1\}$ if $s > 0$.) Thus we can determine the best branch-i match of a (if any) in reach(χ, τ) in $O(\ell_i \log \ell_i)$ time, and hence we can determine the best match of a in reach(χ, τ) in $O(n \log n)$ time.

We now describe Subroutine 2. The input to Subroutine 2 is a nice pair (χ, τ) for F and an integer i in $[m]$ such that $s < \ell_i$ where $j_1 < \cdots < j_s$ denotes indices(χ, τ, i). The output k of Subroutine 2 is the maximum j such that $\langle i, j\rangle$ is a possible match of agent $a = \langle i, s+1\rangle$ in reach(χ, τ), or 0 if no such j exists. As in Subroutine 1, we can reduce this task to a path problem. In the present case, we can restrict χ and τ to branch i, that is, we do not need to consider the extended path that includes the center object. Moreover, if $s > 0$ we can restrict the binary search for j to the interval $\{j_s + 1, \ldots, \ell_i\}$. Each iteration of the binary search involves a single call to Algorithm 1. Given the results of the preprocessing phase, the additional time complexity required for each such call is $O(\ell_i)$. Taking into account the binary search, this approach yields a time complexity of $O(\ell_i \log \ell_i)$.

Having discussed Subroutines 1 and 2, we can now establish an upper

bound on the time complexity of the first phase. The first phase performs at most n iterations. The worst-case running time of each iteration is dominated by the cost of a possible call to Subroutine 1, and hence is $O(n \log n)$. Thus the overall running time of the first phase is $O(n^2 \log n)$.

We now discuss Subroutine 3 and the time complexity of the second phase. Since $\tau(\chi(o)) = o$ throughout the second phase, the sequence of calls to Subroutine 3 that we make for a given value of i can be resolved by restricting attention to the branch-i objects and their matched agents under configuration χ_1. Because (χ_1, τ) is a nice pair for F at the end of the first phase, and because we iterate over increasing values of j from $s+1$ to ℓ_i, the approach of Section 2.3.4 can be used to implement each successive call to Subroutine 3 in $O(\ell_i)$ time. Thus the time required to process branch i is $O(\ell_i^2)$, and the overall time complexity of the second phase is $\sum_{i \in [m]} O(\ell_i^2) = O(n^2)$.

Since the time complexity of the preprocessing phase is $O(n^2)$, the time complexity of the first phase is $O(n^2 \log n)$, and the time complexity of the second phase is $O(n^2)$, we conclude that the overall time complexity of the algorithm is $O(n^2 \log n)$.

2.4.2 The First Phase Invariant

The following sequence of lemmas pertain to an arbitrary iteration in the first phase. We assume that invariant I holds before the iteration, and we seek to prove that invariant I holds after the iteration. We use the symbols σ, χ, and τ (resp., σ', χ', τ') to refer to the values of the corresponding program

variables at the start (resp., end) of the iteration. We use the symbols i, j_0, s, $j_1, \ldots, j_s$, a, μ, μ^*, and χ^* to refer to the corresponding program variables; these variables do not change in value during the iteration. We use the symbol k to refer to the initial value of the corresponding program variable. The value of the program variable k can change at most once (see step 4(c)). We use the symbol k' to refer to the value of program variable k at the end of the iteration.

Lemma 2.4.1. *We have $j_t > t$ for all t in $[s]$.*

Proof. Since (χ, τ) is nice, we have $\text{reach}(\chi, \tau) \neq \emptyset$. Let μ' denote a matching in $\text{reach}(\chi, \tau)$. Since $\tau(\chi(\langle i, 1\rangle)) = \langle i, j_1\rangle$ and agent $\chi(o)$ cannot overtake agent $\chi(\langle i, 1\rangle)$ as we transform the matching associated with χ to μ', we have $1 \leq j_0 < j_1$. Since $j_1 < \cdots < j_s$, we deduce that $j_2 > 2$, $\ldots$, $j_s > s$. □

It follows from Lemma 2.4.1 that $s < \ell_i$ and hence that agent a is well-defined. Agent a serves as the dictator in this iteration.

Throughout the remainder of this section, let R denote the set of all matchings in $\text{reach}(\chi, \tau)$ that match a to an object in branch i, and let J denote the set of all integers j in $[\ell_i]$ such that object $\langle i, j\rangle$ is a possible match of agent a in R. Thus k is the maximum integer in J, or 0 if J is empty.

Lemma 2.4.2. *The following statements hold: (1) if $\langle i, s+1\rangle \prec_a \cdots \prec_a \langle i, 1\rangle \prec_a o$ does not hold, then $\text{reach}(\chi, \tau) = R$; (2) if j belongs to J and $s = 0$, then $j > 1$; (3) if j belongs to J and $s > 0$ then $j > j_s$; (4) if $k > 0$ then $\langle i, k\rangle$ is*

the best match of agent a in R; (5) if $k = 0$ then $\langle i, s+1\rangle \prec_a \cdots \prec_a \langle i, 1\rangle \prec_a o$ holds.

Proof. We begin by proving part (1). Assume that $\langle i, s+1\rangle \prec_a \cdots \prec_a \langle i, 1\rangle \prec_a o$ does not hold. It follows agent a cannot be matched to an object outside of branch i in reach(χ, τ). Hence reach$(\chi, \tau) = R$, as required.

To establish parts (2) through (5), we consider two cases.

Case 1: $J = \emptyset$. In this case, $k = 0$ and hence parts (2), (3), and (4) hold vacuously. It remains to prove part (5). Assume for the sake of contradiction that $\langle i, s+1\rangle \prec_a \cdots \prec_a \langle i, 1\rangle \prec_a o$ does not hold. It follows from part (1) that reach$(\chi, \tau) = R = \emptyset$, contradicting invariant I.

Case 2: $J \neq \emptyset$. Thus $k > 0$ and hence part (5) holds vacuously. In the proofs of parts (2), (3), and (4) below, let j belong to J, and let μ' be a matching in R such that $\mu'(a) = \langle i, j\rangle$.

We first argue that part (2) holds. Assume for the sake of contradiction that $s = 0$ and $j = 1$. Thus $\mu'(a) = \chi(a) = \langle i, 1\rangle$, and hence $j_0 > 1$. It follows that agent $\chi(o)$ overtakes the stationary agent a as we transform the matching associated with χ to μ', a contradiction.

Now we argue that part (3) holds. Assume that $s > 0$. We begin by proving that $j \geq s + 1$. Assume for the sake of contradiction that $j < s + 1$. Since (χ, τ) is nice, a is the closest branch-i agent to the center that is "inward-moving" in the sense that $\mu'(a)$ is closer to the center than $\chi(a)$. Since no inward-moving agent can overtake another inward-moving agent, and since

agent $\chi(o)$ moves into branch i, we deduce that agent a moves out of branch i, a contradiction since j belongs to J. Thus $j \geq s+1$. Now we argue that $j > j_s$. Lemma 2.4.1 implies that agent $\chi(\langle i, s\rangle)$ moves outward to object $\langle i, j_s\rangle$. Since $j \geq s+1$, agent a is either stationary or outward-moving, and hence cannot be overtaken by the outward-moving agent $\chi(\langle i, s\rangle)$. It follows that $j > j_s$, as required.

Now we argue that part (4) holds. Since $j > j_s$ and Lemma 2.4.1 implies $j_s > s$, we have $j > s+1$. Thus agent a is outward-moving on branch i. It follows that $\langle i, k\rangle$ is the best match of agent a in R, as required. □

Lemma 2.4.3. *Assume that* $\langle i, s+1\rangle \prec_a \cdots \prec_a \langle i, 1\rangle \prec_a o$ *holds. Then* $\mu \rightsquigarrow_F \mu^*$ *and*

$$\text{reach}(\chi, \tau) \setminus R = \text{reach}(\chi^*, \tau) \neq \emptyset.$$

Proof. Since (χ, τ) is nice and $\langle i, s+1\rangle \prec_a \cdots \prec_a \langle i, 1\rangle \prec_a o$ holds, Lemma 2.4.1 implies that the $s+1$ exchanges used to transform μ to μ^* are all Pareto-improving. Hence $\mu \rightsquigarrow_F \mu^*$,

The set $\text{reach}(\chi^*, \tau)$ is nonempty since it includes μ^*. It remains to prove that $\text{reach}(\chi, \tau) \setminus R = \text{reach}(\chi^*, \tau)$.

We first argue that $\text{reach}(\chi^*, \tau) \subseteq \text{reach}(\chi, \tau) \setminus R$. Let μ^{**} belong to $\text{reach}(\chi^*, \tau)$. Thus $\mu^* \rightsquigarrow_F \mu^{**}$. Since $\mu \rightsquigarrow_F \mu^*$, we conclude that $\mu \rightsquigarrow_F \mu^{**}$ and hence μ^{**} belongs to $\text{reach}(\chi, \tau)$. Since $\chi^*(o) = a$ and $\langle i, 1\rangle \prec_a o$, we deduce that $\mu^{**}(a)$ does not belong to branch i. Since μ^{**} belongs to

reach(χ, τ) and $\mu^{**}(a)$ does not belong to branch i, we conclude that μ^{**} belongs to reach$(\chi, \tau) \setminus R$.

Now we argue that reach$(\chi, \tau) \setminus R \subseteq$ reach(χ^*, τ). Let μ^{**} belong to reach$(\chi^*, \tau) \setminus R$. From our discussion of the reachable matching problem on trees in Section 2.7.1, we can obtain a sequence of swaps that transforms μ to μ^{**} by repeatedly performing any swap that moves the two participating agents closer to their matched objects under μ^{**}. Since $\mu^{**}(a)$ does not belong to branch i, this means that we can begin by performing the $s+1$ swaps that transform μ to μ^*. It follows that μ^{**} belongs to reach(χ^*, τ), as required. □

Lemma 2.4.3 implies that if line 4(b) is executed, then reach$(\chi^*, \tau) \neq \emptyset$, and hence object b is well-defined.

Lemma 2.4.4. *Assume that* $\langle i, s+1\rangle \prec_a \cdots \prec_a \langle i, 1\rangle \prec_a o$ *holds. If* $k = 0$ *or* $b \succ_a \langle i, k\rangle$*, then* b *is the best match of agent* a *in* reach(χ, τ) *and*

$$\text{reach}(\chi, \tau') = \text{reach}(\chi', \tau').$$

Otherwise, $\langle i, k\rangle$ *is the best match of agent* a *in* reach(χ, τ).

Proof. We consider two cases.

Case 1: $k = 0$. In this case, $R = \emptyset$ and $\chi' = \chi^*$. Hence Lemma 2.4.3 implies reach$(\chi^*, \tau) =$ reach(χ, τ). Thus object b is the best match of agent a in reach(χ, τ) and reach$(\chi, \tau') =$ reach(χ', τ').

Case 2: $k > 0$. Since $k > 0$, object $\langle i, k\rangle$ is the best match of agent a in R. Lemma 2.4.3 implies that b is the best match of agent a in $\text{reach}(\chi^*, \tau) = \text{reach}(\chi, \tau) \setminus R$. We consider two subcases.

Case 2.1: $b \succ_a \langle i, k\rangle$. Thus b is the best match of agent a in $\text{reach}(\chi, \tau)$, $\chi' = \chi^*$, $\tau' = \tau + (a, b)$, and $\text{reach}(\chi, \tau') = \text{reach}(\chi', \tau')$.

Case 2.2: $b \prec_a \langle i, k\rangle$. Thus object $\langle i, k\rangle$ is the best match of agent a in $\text{reach}(\chi, \tau)$. □

The next lemma establishes that invariant I holds at the end of the iteration.

Lemma 2.4.5. *At the end of the iteration,* (χ', τ') *is a nice pair for* F*, agent* $\chi'(o)$ *is matched in* τ'*,* $\text{reach}(\chi', \tau') = \text{reach}(\chi_0, \tau')$*, and* $\tau' = \text{serial}(\sigma')$.

Proof. Observe that either $k' = -1$ or $k' = k$. Below we consider these two cases separately.

Case 1: $k' = -1$. In this case, step 4(c) is executed and the associated if condition evaluates to true. Furthermore, the if condition associated with step 5 evaluates to false. Thus $\chi' = \chi^*$ and $\tau' = \tau + (a, b)$. It is straightforward to verify that the pair (χ', τ') is nice for F with $\text{indices}(\chi', \tau', i)$ equal to $j_0 < \cdots < j_s$, and that $\chi'(o) = a$ is matched in τ'. By Lemma 2.4.4, object b is the best match of agent a in $\text{reach}(\chi, \tau)$. Since invariant I implies $\tau = \text{serial}(\sigma)$ and $\text{reach}(\chi, \tau) = \text{reach}(\chi_0, \tau)$, we deduce that $\tau' = \text{serial}(\sigma')$. By Lemma 2.4.4, $\text{reach}(\chi, \tau') = \text{reach}(\chi', \tau')$. Invariant I implies

reach(χ, τ) = reach(χ_0, τ) and hence reach(χ, τ') = reach(χ_0, τ'). We conclude that reach(χ', τ') = reach(χ_0, τ'), as required.

Case 2: $k' = k$. We begin by proving that $k > 0$. Assume for the sake of contradiction that $k = 0$. Part (5) of Lemma 2.4.2 implies that $\langle i, s+1\rangle \prec_a \cdots \prec_a \langle i, 1\rangle \prec_a o$ holds. Hence step 4(c) is executed, and since $k = 0$, the associated if condition evaluates to true. Hence $k' = -1$, contradicting the Case 2 assumption.

Since $k' = k > 0$, we deduce that $\chi' = \chi$ and $\tau' = \tau + (a, \langle i, k\rangle)$. It is straightforward to verify that (χ', τ') is nice for F with indices(χ', τ', i) equal to $j_1 < \cdots < j_s < k$. Since $\chi' = \chi$ and invariant I holds, we deduce that agent $\chi'(o)$ is matched in τ'.

We claim that $\langle i, k\rangle$ is the best match of agent a in reach(χ, τ). If $\langle i, s+1\rangle \prec_a \cdots \prec_a \langle i, 1\rangle \prec_a o$ does not hold, the claim follows from parts (1) and (4) of Lemma 2.4.2, so assume that $\langle i, s+1\rangle \prec_a \cdots \prec_a \langle i, 1\rangle \prec_a o$ holds. Thus the if condition of step 4 evaluates to true, and hence the if condition of step 4(c) is evaluated. Since $k' = k > 0$, the latter if condition evaluates to false, and hence Lemma 2.4.4 implies that the claim holds.

Invariant I implies reach(χ, τ) = reach(χ_0, τ) and τ = serial(σ). Thus the claim of the preceding paragraph implies τ' = serial(σ').

Since $\chi' = \chi$, invariant I implies reach(χ', τ) = reach(χ_0, τ) and hence reach(χ', τ') = reach(χ_0, τ'). □

2.5 NP-Completeness of Reachable Object on Cliques

It is easy to see that the reachable object on cliques problem belongs to NP. In this section, we prove that the problem is NP-hard by presenting a polynomial-time reduction from the known NP-complete problem 2P1N-SAT to reachable object on cliques.

An instance of 2P1N-SAT is a propositional formula f over n variables $x_1, \ldots, x_n$ with the following properties: f is the conjunction of a number of clauses, where each clause is the disjunction of a number of literals, and each literal is either a variable or the negation of a variable; each variable occurs exactly three times in f, once in each of three distinct clauses; each variable x_i occurs twice as a positive literal (i.e., x_i) and once as a negative literal (i.e., $\neg x_i$).

Throughout the remainder of Section 2.5, let f denote a given instance of 2P1N-SAT with n variables $x_1, \ldots, x_n$ and m clauses $C_1, \ldots, C_m$.

In Section 2.5.1, we describe a polynomial-time procedure for transforming f into an instance I of reachable object on cliques. In Section 2.5.2, we prove that f is a positive instance of 2P1N-SAT if and only if I is a positive instance of reachable object on cliques.

2.5.1 Description of the Reduction

We now describe how we transform a 2P1N-SAT instance f into a corresponding instance I of reachable object on cliques. For each variable x_i

in f, there are two agents $\hat{x}_i^1$ and $\hat{x}_i^2$ in I. For each clause C_j in f, there are three agents $\hat{u}_j, \hat{v}_j$ and $\hat{w}_j$ in I. Note that the name we use to refer to each agent in I includes a hat symbol. We adopt the convention that if the hat symbol is removed from the name of such an agent, we obtain the name of the initial endowment of that agent. For example, agents $\hat{u}_j$ and $\hat{x}_i^1$ are initially endowed with objects u_j and x_i^1, respectively. For convenience, we define $\hat{U}$ as the set of agents $\{\hat{u}_j \mid j \in [m]\}$, and we define U as the set of objects $\{u_j \mid j \in [m]\}$. We define $\hat{V}$, V, $\hat{W}$, W, $\hat{X}$, and X similarly.

Let A (resp., B) denote the set of all agents (resp., objects) in I. Let N denote $|B|$. Thus $N = 3m + 2n$. Let K_N denote the complete graph with vertex set B, and let E denote the edge set of K_N. Let μ_0 denote the initial matching of agents with objects.

Below we describe the preferences $\succ$ of the agents in A over the objects in B. Let $\chi = (F, \mu_0)$ denote the initial configuration of I, where $F = (A, B, \succ, E)$. Instance I asks the following reachability question: Is there a matching μ in reach(χ) such that $\mu(\hat{w}_m) = u_1$?

Let variable x_i appear in clauses $C_{p_i^1}$ and $C_{p_i^2}$ as a positive literal, where $p_i^1 < p_i^2$, and in clause C_{n_i} as a negative literal. The definition of 2P1N-SAT implies that p_i^1, p_i^2, and n_i are distinct.

Below we list the preferences of each agent in A. In doing so, we specify only the objects that an agent prefers to its initial endowment; the order of the remaining objects is immaterial. The initial endowment is shown in a box.

For any i in $[n]$, the preferences of agent $\hat{x}_i^1$ are

$$\hat{x}_i^1 : x_i^2 \succ w_{p_i^1} \succ v_{p_i^1} \succ \boxed{x_i^1}$$

and the preferences of agent $\hat{x}_i^2$ are

$$\hat{x}_i^2 : w_{n_i} \succ v_{n_i} \succ w_{p_i^2} \succ v_{p_i^2} \succ x_i^1 \succ \boxed{x_i^2}$$

if $n_i < p_i^2$, and are

$$\hat{x}_i^2 : w_{p_i^2} \succ v_{p_i^2} \succ w_{n_i} \succ v_{n_i} \succ x_i^1 \succ \boxed{x_i^2}$$

otherwise.

For any j in $[m]$, the preferences of agent $\hat{u}_j$ are

$$\hat{u}_j : v_j \succ \boxed{u_j}.$$

For any j in $[m-1]$, the preferences of agent $\hat{w}_j$ are

$$\hat{w}_j : u_{j+1} \succ \boxed{w_j}.$$

The preferences of agent $\hat{w}_m$ are

$$\hat{w}_m : u_1 \succ v_1 \succ w_1 \succ u_2 \succ v_2 \succ w_2 \succ \ldots \succ u_m \succ v_m \succ \boxed{w_m}.$$

For any j in $[m]$, the preferences of agent $\hat{v}_j$ are

$$\hat{v}_j : \{x_i^1 \mid j \in \{p_i^1, n_i\}\} \cup \{x_i^2 \mid j = p_i^2\} \succ \boxed{v_j},$$

where the set of objects preceding v_j may be ordered arbitrarily.

This completes the description of the reachable object on cliques instance I.

2.5.2 Correctness of the Reduction

In this section, we prove that f is a positive instance of 2P1N-SAT if and only if I is a positive instance of reachable object on cliques. Lemma 2.5.4 establishes the only if direction. Lemmas 2.5.11 through 2.5.17 lay the groundwork for Lemma 2.5.18, which establishes the if direction.

For the purposes of our analysis, it is convenient to assign a non-negative integer rank to each object in B, as follows. For any j in $[m]$, we define $\text{rank}(u_j)$ as $3j-2$, $\text{rank}(v_j)$ as $3j-1$, and $\text{rank}(w_j)$ as $3j$. The rank of any object in X is defined to be 0.

Observation 2.5.1 below can be justified by enumerating all those agents who like object x_i^1 at least as well as their initial endowment. Observations 2.5.2 and 2.5.3 can be justified in a similar manner.

Observation 2.5.1. *For any i in $[n]$ and any matching μ in* $\text{reach}(\chi)$*, agent $\mu^{-1}(x_i^1)$ belongs to $\{\hat{x}_i^1, \hat{x}_i^2, \hat{v}_{p_i^1}, \hat{v}_{n_i}\}$.*

Observation 2.5.2. *For any i in $[n]$ and any matching μ in* $\text{reach}(\chi)$*, agent $\mu^{-1}(x_i^2)$ belongs to $\{\hat{x}_i^1, \hat{x}_i^2, \hat{v}_{p_i^2}\}$.*

Observation 2.5.3. *For any j in $[m]$ and any matching μ in* $\text{reach}(\chi)$*, agents $\mu^{-1}(v_j)$ and $\mu^{-1}(w_j)$ belong to $\{\hat{u}_j, \hat{v}_j, \hat{w}_m\} \cup A_j$ and $\{\hat{w}_j, \hat{w}_m\} \cup A_j$, respectively, where A_j denotes*

$$\{\hat{x}_i^1 \mid j = p_i^1\} \cup \{\hat{x}_i^2 \mid j \in \{p_i^2, n_i\}\}.$$

Lemma 2.5.4. *Assume that 2P1N-SAT instance f is satisfiable. Then there is a matching μ in* reach(χ) *such that $\mu(\hat{w}_m) = u_1$ in the reachable object on cliques instance I.*

Proof. Let $\sigma : \{x_1, \ldots, x_n\} \to \{0, 1\}$ denote a satisfying assignment for f. We specify a sequence of matchings $\mu_0, \ldots, \mu_{3n+3m-1}$, which depends on σ, such that (1) $\mu_{3n+3m-1}(\hat{w}_m) = u_1$ and (2) $\mu_{k-1} = \mu_k$ or $\mu_{k-1} \to_F \mu_k$ for all k in $[3n + 3m - 1]$. We obtain this sequence in two phases.

In the first phase, we perform the following three steps for each i from 1 to n.

1. If $\sigma(x_i) = 1$ and $\mu_{3i-3}(\hat{v}_{p_i^1}) = v_{p_i^1}$, we set μ_{3i-2} to the matching obtained by swapping $\hat{v}_{p_i^1}$ with $\hat{x}_i^1$ in μ_{3i-3}. Otherwise, we set μ_{3i-2} to μ_{3i-3}.

2. If $\sigma(x_i) = 1$ and $\mu_{3i-2}(\hat{v}_{p_i^2}) = v_{p_i^2}$, we set μ_{3i-1} to the matching obtained by swapping $\hat{v}_{p_i^2}$ with $\hat{x}_i^2$ in μ_{3i-2}. Otherwise, we set μ_{3i-1} to μ_{3i-2}.

3. If $\sigma(x_i) = 0$ and $\mu_{3i-1}(\hat{v}_{n_i}) = v_{n_i}$, we set μ_{3i} to the matching obtained by first swapping $\hat{x}_i^1$ with $\hat{x}_i^2$, and then swapping $\hat{v}_{n_i}$ with $\hat{x}_i^2$, in μ_{3i-1}. Otherwise, we set μ_{3i} to μ_{3i-1}.

It is easy to check that all of the swaps in the first phase are valid.

Let A_j be as defined in the statement of Observation 2.5.3. We claim that at the end of the first phase, agent $\mu_{3n}^{-1}(v_j)$ belongs to A_j for all j in $[m]$. Assume for the sake of contradiction that for some j in $[m]$, agent $\mu_{3n}^{-1}(v_j)$ does

not belong to A_j. Since agents $\hat{u}_j$ and $\hat{w}_m$ do not participate in any swap in the first phase, $\mu_{3n}(\hat{u}_j) = u_j$ and $\mu_{3n}(\hat{w}_m) = w_m$. Since $\mu_{3n}^{-1}(v_j)$ does not belong to $\{\hat{u}_j, \hat{w}_m\} \cup A_j$, Observation 2.5.3 implies that $\mu_{3n}(\hat{v}_j) = v_j$. Let variable x_i satisfy clause C_j. The swaps in the ith iteration of first phase imply that $\mu_{3i}^{-1}(v_j) \neq \hat{v}_j$. Since no agent in $\hat{V}$ participates in more than one swap in the first phase, we have $\mu_{3n}^{-1}(v_j) \neq \hat{v}_j$, a contradiction since $\mu_{3n}(\hat{v}_j) = v_j$. This completes the proof of the claim.

Note that there is a unique object of rank k for each k in $[3m]$. Using the claim stated above, together with the fact that no agent in $\hat{U} \cup \hat{W}$ participates in a swap in the first phase, it is easy to verify that for any rank k in $[3m-1]$, there is an agent a such that $\text{rank}(\mu_{3n}(a)) = k$ and a prefers the object of rank $k+1$ to the object of rank k. The preferences of agent $\hat{w}_m$ imply that for any rank k in $[3m-1]$, agent $\hat{w}_m$ prefers the object of rank k to the object of rank $k+1$. Moreover, $\text{rank}(\mu_{3n}(\hat{w}_m)) = 3m$. In the second phase we perform $3m-1$ swaps, each involving agent $\hat{w}_m$. For k running from $3m-1$ down to 1, we set $\mu_{3n+3m-k}$ to the matching obtained by swapping $\hat{w}_m$ with the agent matched to the object of rank k in $\mu_{3n+3m-k-1}$. The foregoing arguments show that all of these $3m-1$ swaps are valid and $\text{rank}(\mu_{3n+3m-1}(\hat{w}_m)) = 1$. Thus $\mu_{3n+3m-1}(\hat{w}_m) = u_1$. □

Observation 2.5.5 below can be justified by using the preferences of agent $\hat{w}_m$ and the fact that if any agent a swaps from object b to b' then $b' \succ_a b$. Observations 2.5.6 through 2.5.10 can be justified similarly.

Observation 2.5.5. *For any matchings μ_1 and μ_2 in* $\text{reach}(\chi)$ *such that $\mu_1 \to_F \mu_2$, we have* $\text{rank}(\mu_2(\hat{w}_m)) \leq \text{rank}(\mu_1(\hat{w}_m))$.

Observation 2.5.6. *For any j in $[m-1]$ and any matchings μ_1 and μ_2 in* $\text{reach}(\chi)$ *such that $\mu_1 \to_F \mu_2$, we have* $\text{rank}(\mu_2(\hat{w}_j)) \leq \text{rank}(\mu_1(\hat{w}_j)) + 1$.

Observation 2.5.7. *For any j in $[m]$ and any matchings μ_1 and μ_2 in* $\text{reach}(\chi)$ *such that $\mu_1 \to_F \mu_2$, we have* $\text{rank}(\mu_2(\hat{u}_j)) \leq \text{rank}(\mu_1(\hat{u}_j)) + 1$.

Observation 2.5.8. *For any j in $[m]$ and any matchings μ_1 and μ_2 in* $\text{reach}(\chi)$ *such that $\mu_1 \to_F \mu_2$, we have* $\text{rank}(\mu_2(\hat{v}_j)) \leq \text{rank}(\mu_1(\hat{v}_j))$.

Observation 2.5.9. *For any i in $[n]$ and any matchings μ_1 and μ_2 in* $\text{reach}(\chi)$ *such that $\mu_1 \to_F \mu_2$ and $\mu_1(\hat{x}_i^1)$ belongs to $B \setminus X$, we have* $\text{rank}(\mu_2(\hat{x}_i^1)) \leq \text{rank}(\mu_1(\hat{x}_i^1)) + 1$.

Observation 2.5.10. *For any i in $[n]$ and any matchings μ_1 and μ_2 in* $\text{reach}(\chi)$ *such that $\mu_1 \to_F \mu_2$ and $\mu_1(\hat{x}_i^2)$ belongs to $B \setminus X$, we have* $\text{rank}(\mu_2(\hat{x}_i^2)) \leq \text{rank}(\mu_1(\hat{x}_i^2)) + 1$.

Lemma 2.5.11 below follows from Observations 2.5.5 through 2.5.10.

Lemma 2.5.11. *For any agent a in A and any matchings μ_1 and μ_2 in* $\text{reach}(\chi)$ *such that $\mu_1 \to_F \mu_2$ and $\mu_1(a)$ belongs to $B \setminus X$, we have* $\text{rank}(\mu_2(a)) \leq \text{rank}(\mu_1(a)) + 1$.

Lemma 2.5.12. *Let μ_1 and μ_2 be matchings in* $\text{reach}(\chi)$ *such that $\mu_1 \to_F \mu_2$. Then*

$$\text{rank}(\mu_2(\hat{w}_m)) \geq \text{rank}(\mu_1(\hat{w}_m)) - 1.$$

Proof. Assume for the sake of contradiction that $\text{rank}(\mu_2(\hat{w}_m)) < \text{rank}(\mu_1(\hat{w}_m)) - 1$. The preferences of $\hat{w}_m$ imply that $\mu_1(\hat{w}_m)$ and $\mu_2(\hat{w}_m)$ belong to $B \setminus X$. Thus there is an agent a such that $\mu_1(a)$ belongs to $B \setminus X$ and $\text{rank}(\mu_2(a)) > \text{rank}(\mu_1(a)) + 1$, contradicting Lemma 2.5.11. □

Lemma 2.5.13. *Let i belong to $[n]$ and j belong to $[m]$. If $\hat{v}_j$ prefers x_i^1 to v_j, then $\hat{v}_j$ prefers v_j to x_i^2. Similarly, if $\hat{v}_j$ prefers x_i^2 to v_j, then $\hat{v}_j$ prefers v_j to x_i^1.*

Proof. Assume that $\hat{v}_j$ prefers x_i^1 to v_j. The preferences of $\hat{v}_j$ imply that $j \in \{p_i^1, n_i\}$. Since p_i^1, p_i^2, and n_i are distinct, $j \neq p_i^2$. Hence the preferences of $\hat{v}_j$ imply that $\hat{v}_j$ prefers v_j to x_i^2. We can use a similar argument to prove that if $\hat{v}_j$ prefers x_i^2 to v_j, then $\hat{v}_j$ prefers v_j to x_i^1. □

Lemma 2.5.14. *Let μ_1 and μ_2 be matchings in* $\text{reach}(\chi)$ *such that $\mu_1 \to_F \mu_2$ and $\mu_1(\hat{x}_i^2) = x_i^2$. Then $\mu_2(\hat{x}_i^2)$ belongs to $\{x_i^1, x_i^2, v_{p_i^2}\}$.*

Proof. By examining the preferences of $\hat{x}_i^2$ we deduce that object $\mu_2(\hat{x}_i^2)$ belongs to

$$\{w_{n_i}, w_{p_i^2}, v_{n_i}, v_{p_i^2}, x_i^1, x_i^2\}.$$

Assume for the sake of contradiction that $\mu_2(\hat{x}_i^2)$ belongs to $\{w_{n_i}, w_{p_i^2}, v_{n_i}\}$. We consider three cases.

Case 1: $\mu_2(\hat{x}_i^2) = w_{n_i}$. Thus $\mu_1^{-1}(w_{n_i}) = \mu_2^{-1}(x_i^2)$. Since $\mu_2^{-1}(x_i^2) \neq \hat{x}_i^2$, Observation 2.5.2 implies that $\mu_2^{-1}(x_i^2)$ belongs to $\{\hat{x}_i^1, \hat{v}_{p_i^2}\}$. We consider two cases.

Case 1.1: $\mu_2^{-1}(x_i^2) = \hat{x}_i^1$. Thus $\mu_1^{-1}(w_{n_i}) = \hat{x}_i^1$. By examining the preferences of $\hat{x}_i^1$, we deduce that $\mu_1(\hat{x}_i^1) \neq w_{n_i}$, a contradiction.

Case 1.2: $\mu_2^{-1}(x_i^2) = \hat{v}_{p_i^2}$. A contradiction follows using a similar argument as in Case 1.1.

Case 2: $\mu_2(\hat{x}_i^2) = w_{p_i^2}$. A contradiction follows using a similar argument as in Case 1.

Case 3: $\mu_2(\hat{x}_i^2) = v_{n_i}$. A contradiction follows using a similar argument as in Case 1.

Thus $\mu_2(\hat{x}_i^2)$ belongs to $\{x_i^1, x_i^2, v_{p_i^2}\}$. □

Throughout the remainder of Section 2.5.2, we say that an agent a is satisfied in a matching μ if $\mu(a)$ is the most preferred object of a. In Lemmas 2.5.15 through 2.5.18 below, let $\mu_0, \ldots, \mu_t$ be a sequence of matching such that $\mu_{s-1} \rightarrow_F \mu_s$ for all s in $[t]$, and for each i in $[n]$ let $P(i)$, (resp., $Q(i)$ and $R(i)$) denote the predicate that holds if there is an integer s in $[t]$ such that $\mu_s(\hat{x}_i^1) = v_{p_i^1}$ (resp., $\mu_s(\hat{x}_i^2) = v_{p_i^2}$, $\mu_s(\hat{x}_i^2) = v_{n_i}$). Lemmas 2.5.15 and 2.5.16 below present useful properties of these predicates.

Lemma 2.5.15. *Let i be an element of $[n]$ such that $R(i)$ holds. Then $Q(i)$ does not hold.*

Proof. Let s be an element of $[t]$ such that $\mu_s(\hat{x}_i^2) = v_{n_i}$; such an s exists since $R(i)$ holds. Assume that $Q(i)$ holds. Let s' be an element of $[t]$ such that $\mu_{s'}(\hat{x}_i^2) = v_{p_i^2}$; such an s' exists as $Q(i)$ holds. We consider two cases.

Case 1: $s' > s$. Let s'' be an integer such that $s \leq s'' \leq s'$. The preferences of agent $\hat{x}_i^2$ imply that $p_i^2 < n_i$ and $\mu_{s''}(\hat{x}_i^2)$ belongs to $\{w_{p_i^2}, v_{p_i^2}, w_{n_i}, v_{n_i}\}$. Hence $\text{rank}(\mu_{s''}(\hat{x}_i^2))$ belongs to $\{3p_i^2, 3p_i^2-1, 3n_i, 3n_i-1\}$. Note that $\text{rank}(\mu_s(\hat{x}_i^2)) = 3n_i-1$, and $\text{rank}(\mu_{s'}(\hat{x}_i^2)) = 3p_i^2-1$. Hence there is an s''' such that $s \leq s''' < s'$ and $\text{rank}(\mu_{s'''+1}(\hat{x}_i^2)) \leq \text{rank}(\mu_{s'''}(\hat{x}_i^2)) - 2$. It follows that there is an agent a such that $\mu_{s'''}(a)$ belongs to $B \setminus X$ and $\text{rank}(\mu_{s'''+1}(a)) \geq \text{rank}(\mu_{s'''}(a)) + 2$, contradicting Lemma 2.5.11.

Case 2: $s' < s$. We can derive a contradiction using a similar argument as in Case 1. □

Lemma 2.5.16. *Let i be an element of $[n]$ such that $R(i)$ holds. Then $P(i)$ does not hold.*

Proof. Let s be an element of $[t]$ such that $\mu_s(\hat{x}_i^2) = v_{n_i}$; such an s exists since $R(i)$ holds. We begin by proving the following claim: There is an integer s'' in $[s-1]$ such that $\mu_{s''}(\hat{x}_i^2) = x_i^1$. Assume for the sake of contradiction that there is no s'' in $[s-1]$ such that $\mu_{s''}(\hat{x}_i^2) = x_i^1$. Let s'' be the least index in $[s]$ such that $\mu_{s''}(\hat{x}_i^2) \neq x_i^2$. Since $\mu_{s''}(\hat{x}_i^2)$ does not belong to $\{x_i^1, x_i^2\}$, Lemma 2.5.14 implies that $\mu_{s''}(\hat{x}_i^2) = v_{p_i^2}$. Thus $Q(i)$ holds, contradicting Lemma 2.5.15. This completes the proof of the claim.

Having established the claim, we let s'' denote the least integer in $[s-1]$ such that $\mu_{s''}(\hat{x}_i^2) = x_i^1$. The preferences of agent $\hat{x}_i^2$ imply that $\mu_{s''-1}(\hat{x}_i^2) = x_i^2$. Let a be the agent $\mu_{s''-1}^{-1}(x_i^1)$. Since $a \neq \hat{x}_i^2$, Observation 2.5.1 implies that a belongs to $\{\hat{x}_i^1, \hat{v}_{p_i^1}, \hat{v}_{n_i}\}$. We consider two cases.

Case 1: $a \in \{\hat{v}_{p_i^1}, \hat{v}_{n_i}\}$. Lemma 2.5.13 implies that a does not prefer x_i^2 to their initially endowment. Hence $\mu_{s''}^{-1}(x_i^2) \neq a$, a contradiction.

Case 2: $a = \hat{x}_i^1$. Since $\mu_0(\hat{x}_i^1) = \mu_{s''-1}(\hat{x}_i^1) = x_i^1$, we deduce that $\mu_s(\hat{x}_i^1) = x_i^1 \neq v_{p_i^1}$ for all s' such that $0 \leq s' < s''$. Moreover, $\hat{x}_i^1$ is satisfied in $\mu_{s''}$ and hence $\mu_{s'}(\hat{x}_i^1) = x_i^2 \neq v_{p_i^1}$ for all s' such that $s'' \leq s' \leq t$. Hence $\mu_{s'}(\hat{x}_i^1) \neq v_{p_i^1}$ for all s' such that $0 \leq s' \leq t$. Thus $P(i)$ does not hold. □

Lemma 2.5.17. *Let j belong to $[m]$ and assume that $\mu_t(\hat{w}_m) = u_1$. Then there is an s in $[t-1]$ such that $\mu_s(\hat{w}_m) = w_j$ and $\mu_{s+1}(\hat{w}_m) = v_j$.*

Proof. The only object with rank $3j$ (resp., $3j-1$) is w_j (resp., v_j). Since $\text{rank}(\mu_0(\hat{w}_m)) = 3m$ and $\text{rank}(\mu_t(\hat{w}_m)) = 1$, Lemma 2.5.12 implies that for every rank k in $[3m-1]$, there is an integer s in $[t-1]$ such that $\text{rank}(\mu_s(\hat{w}_m)) = k+1$ and $\text{rank}(\mu_{s+1}(\hat{w}_m)) = k$. The lemma follows by choosing k to be $3j-1$. □

Lemma 2.5.18. *Assume that $\mu_t(\hat{w}_m) = u_1$. Then the 2P1N-SAT instance f is satisfiable.*

Proof. We construct an assignment $\sigma : \{x_1, \ldots, x_n\} \to \{0,1\}$ for f as follows: for any i in $[n]$, we set $\sigma(x_i)$ to 1 if $P(i) \vee Q(i)$ holds, and to 0 otherwise.

We now show that σ satisfies f. Let j belong to $[m]$. Let s be an element of $[t-1]$ such that $\mu_s(\hat{w}_m) = w_j$ and $\mu_{s+1}(\hat{w}_m) = v_j$; such an s exists by Lemma 2.5.17. Thus there is an agent a such that $\mu_s(a) = v_j$ and $\mu_{s+1}(a) = w_j$. Since $a \neq \hat{w}_m$, Observation 2.5.3 implies that exactly one of the

following three statements holds: (1) $j = p_i^1$ and $\mu_s(\hat{x}_i^1) = v_j$; (2) $j = p_i^2$ and $\mu_s(\hat{x}_i^2) = v_j$; (3) $j = n_i$ and $\mu_s(\hat{x}_i^2) = v_j$. We consider two cases.

Case 1: (1) or (2) holds. Then $P(i)$ or $Q(i)$ holds, respectively. Hence $\sigma(x_i) = 1$. By construction, the variable x_i appears as the positive literal x_i in clause C_j. Thus, C_j is satisfied.

Case 2: (3) holds. Then $R(i)$ holds. Lemmas 2.5.15 and 2.5.16 imply that $P(i)$ and $Q(i)$ do not hold. Hence $\sigma(x_i) = 0$. By construction, the variable x_i appears as the negative literal $\neg x_i$ in clause C_j. Thus, C_j is satisfied.

Since σ satisfies each clause C_j in f, σ satisfies f. □

Theorem 2.5.19. *Reachable object on cliques is NP-complete.*

Proof. In Section 2.5.1, we described a polynomial-time reduction from 2P1N-SAT instance f to reachable object on cliques instance I. Thus the theorem follows from Lemmas 2.5.4 and 2.5.18. □

2.6 Other NP-Completeness and NP-Hardness Results

We prove the remaining NP-completeness and NP-hardness results stated in Table 2.1. Specifically, we show that the reachable object problem on generalized stars is NP-complete, reachable matching problem on cliques is NP-complete, and Pareto-efficient matching problem on cliques is NP-hard. These results are proved by adapting the corresponding proofs of Gourvès et al. [91] and Müller and Bentert [130] for the object-moving model. The most significant changes are associated with the proof of the first result.

2.6.1 NP-Completeness for Reachable Object on Generalized Stars

Theorem 2.6.1. *Reachable object on generalized stars is NP-complete.*

Proof. It is easy to see that reachable object on generalized stars belongs to NP. We use a reduction from the problem 2P1N-SAT to establish that the reachable object on generalized stars is NP-complete. In an instance of 2P1N-SAT, we are given a propositional formula f that is the conjunction of m clauses $C_1, \ldots, C_m$. Each clause C_i is the disjunction of a number of literals, where each literal is either a variable or the negation of a variable. The set of variables is $x_1, \ldots, x_n$. For each variable x_j, the positive literal x_j appears in exactly two clauses, and the negative literal $\neg x_j$ appears in exactly one clause. We are asked to determine whether the formula f is satisfiable.

Given such a formula f, we construct a corresponding reachable object on generalized stars instance $I = (F, \mu)$ where $F = (A, B, \succ, E)$ as follows. We begin by describing the set of objects B. For each clause index i, there are two objects c'_i and c''_i in B. There is also a special object c''_0 in B. For each variable index j, there are five objects in B: "dummy" objects d_j and d'_j; an object n_j corresponding to the lone occurrence of the negative literal $\neg x_j$ in f; an object p_j corresponding to the first occurrence of the positive literal x_j in f (i.e., the occurrence associated with the lower-indexed clause); an object p'_j corresponding to the other occurrence of the positive literal x_j. Thus there are a total of $2m + 5n + 1$ objects in B.

Observe that our identifier for any given object in B includes a single

lowercase letter. By changing this letter to upper case, we obtain our identifier for the agent initially matched to that object. So, for example, agent C''_0 is initially matched to object c''_0.

We now describe the edge set E. Object c''_m is the center object. We begin by describing $m+n+1$ vertex-disjoint subgraphs of (B, E): a "clause gadget" for each clause index i, a "variable gadget" for each variable index j, and an additional gadget that we call the "staircase". Clause gadget i consists of the lone object c'_i. Variable gadget j is a path of length 3 containing the objects n_j, d_j, p'_j and p_j, in that order, and the lone object d'_j. The staircase is a path of length m containing the sequence of objects $c''_0, c''_1, \ldots, c''_m$. We say that object c''_0 is at the bottom of the staircase, and that object c''_m is at the top of the staircase. The following additional $m+2n$ edges are used to connect these $m+n+1$ subgraphs into a generalized star: there is an edge from object c'_i to object c''_m for each clause index i; there is two edges from object p_j, d'_j to object c''_m, respectively, for each variable index j.

With regard to the agent preferences, it is enough to consider the set objects(I, a) associated with each agent a in A. It is straightforward to verify that we can choose agent preferences so that the following conditions are satisfied. First, for each clause index i, we have objects$(I, C'_i) = \{c'_i\} \cup \{c''_k \mid i-1 \leq k \leq m\}$. Second, for each integer i such that $0 \leq i \leq m$, we have

$$\text{objects}(I, C''_i) = \{c''_k \mid i \leq k \leq m\} \cup \{p_j \mid j \in [n]\} \cup \{p'_j \mid j \in [n]\}.$$

Finally, for each variable index j, the following properties hold: objects(I, D_j)

is equal to $\{d_j, p'_j, p_j, d'_j, c''_m\}$; objects$(I, D'_j) = \{d'_j, p_j, p'_j, c''_m\}$; objects$(I, N_j)$ is equal to $\{n_j, d_j, p'_j, p_j, c''_m, c'_i\}$ where i is the index of the clause that contains the negative literal x_j; objects(I, P_j) is equal to $\{p_j, p'_j, d_j, c''_m, c'_i\}$ where i is the index of the clause that contains the first occurrence of the positive literal x_j; objects(I, P'_j) is equal to $\{p'_j, d_j, n_j, p_j, c''_m, c'_i\}$ where i is the index of the clause that contains the second occurrence of the positive literal x_j.

We claim that agent C''_0 (which starts out at the bottom of the staircase) can reach object c''_m (at the top of the staircase) if and only if f is satisfiable.

We begin by addressing the "if" direction of the claim. Assume that f is satisfiable. Fix a satisfying assignment for f, and for each clause index i, let ℓ_i denote a literal in C_i that is set to true by this satisfying assignment. Let L_i denote the agent that corresponds to literal ℓ_i, as follows: if ℓ_i is the negative literal $\neg x_j$, then L_i is equal to N_j; if ℓ_i is the first occurrence of the positive literal x_j, then ℓ_i is equal to P_j; if ℓ_i is the second occurrence of the positive literal x_j, then ℓ_i is equal to P'_j. Note that the L_i's are all distinct.

For any variable x_j, let $\mathcal{L}_j = \{L_i \mid 1 \leq i \leq m\} \cap \{N_j, P_j, P'_j\}$. Note that if $|\mathcal{L}_j| \geq 2$ then $\mathcal{L}_j = \{P_j, P'_j\}$.

For each clause index i, $1 \leq i \leq m$, let $f(i)$ denote the unique variable index j such that $L_i \in \{N_j, P_j, P'_j\}$.

We now describe how to perform a sequence of swaps that result in agent C''_0 being matched to object c''_m. We perform these swaps in m phases. We will define each phase so that a certain invariant holds. Specifically, we will

ensure that the following conditions hold after k phases have been completed, $0 \leq k \leq m$: (1) the sequence of agents associated with the $m+1$ staircase objects, $c''_0, \ldots, c''_m$ is $C'_1, \ldots, C'_k, C''_0, \ldots, C''_{m-k}$; (2) for any clause index i such that $k < i \leq m$, the agent matched to object c'_i is C'_i; (3) for any variable index j such that $\mathcal{L}_j \not\subseteq \{L_i \mid 1 \leq i \leq k\}$, the agents matched to the objects n_j, d_j, p'_j, and d'_j are N_j, D_j, P'_j, and D'_j, respectively, and the agent matched to object p_j is P_j if $\{L_i \mid 1 \leq i \leq k\} \cap \mathcal{L}_j = \emptyset$ and belongs to $\{C''_{m-i+1} \mid 1 \leq i \leq k\}$ otherwise.

Notice that if we can prove the invariant holds after m phases, then condition (1) of the invariant implies that agent C''_0 is matched to object c''_m, as desired. It is easy to see that the invariant holds at the outset (i.e., after 0 phases). Let k be an integer such that $1 \leq k \leq m$, and assume that the invariant holds after $k-1$ phases. It remains to describe how to implement phase k so that the claimed invariant holds after phase k.

Let j denote $f(k)$. We implement phase k in three stages. In the first stage, we perform swaps within variable gadget j to move agent L_k to the center object c''_m. To see how to do this, consider the following cases. Notice that Condition (2) of the invariant implies the agent matched with the center object c''_m is C''_{m-k+1}.

Case 1: $L_k = N_j$. Thus $\mathcal{L}_j = \{N_j\}$. Condition (3) of the invariant implies that the agents matched to the objects n_j, d_j, p'_j, p_j, and d'_j are N_j, D_j, P'_j, P_j, and D'_j, respectively. Using 8 swaps within agents of variable gadget j and agent C''_{m-k+1}, we can rearrange these six agents so that the

agents matched to the objects n_j, d_j, p'_j, p_j, c''_m, and d'_j are P'_j, P_j, C''_{m-k+1}, D'_j, N_j, and D_j, respectively. Thus agent L_k is matched to the center object c''_m, as required.

Case 2: $L_k = P_j$.Thus $\{P_j\} \subseteq \mathcal{L}_j \subseteq \{P_j, P'_j\}$. Condition (3) of the invariant implies that L_k is matched to object p_j, so swap L_k with C''_{m-k+1} as required.

Case 3: $L_k = P'_j$ and $\mathcal{L}_j = \{P'_j\}$.

Condition (3) of the invariant implies that the agents matched to the objects n_j, d_j, p'_j, p_j, and d'_j are N_j, D_j, P'_j, P_j, and D'_j, respectively. By swapping P'_j with P_j and then with C''_{m-k+1}, we can ensure that L_k is matched to object c''_m, as required.

Case 4: $L_k = P'_j$ and $\mathcal{L}_j = \{P_j, P'_j\}$. Thus $\{L_i \mid 1 \leq i < k\} \cap \mathcal{L}_j = \{P_j\}$. Condition (3) of the invariant implies that the agents matched to the objects n_j, d_j, p'_j, d'_j, and p_j are N_j, D_j, P'_j, D'_j, and an agent in the set $\{C''_{m-i+1} \mid 1 \leq i < k\}$, respectively. By swapping agent P'_j with the agent matched to p_j and then with C''_{m-k+1}, we can ensure that L_k is matched to object c''_m, as required.

At the start of the second stage, the agent matched with the center object c''_m is L_k (due to the first stage), and the agent matched with object c'_k is C'_k (due to condition (2) of the invariant). In the second stage, we use one swap to rearrange these two agents so that the agents matched to the objects c''_m and c'_k are C'_k and L_k.

At the start of the third stage, the sequence of agents associated with the $m+1$ staircase objects $c''_0, \ldots, c''_m$ is

$$C'_1, \ldots, C'_{k-1}, C''_0, \ldots, C''_{m-k}, C'_k,$$

(due to condition (1) of the invariant and the second stage). In the third stage, we perform $m-k+1$ swaps to move agent C'_k down from the top of the staircase to object c''_{k-1}.

It remains to verify that the invariant holds after phase k. The third stage ensures that condition (1) of the invariant holds after phase k. Condition (2) of the invariant holds after phase k because it held before phase k and the only clause gadget involved in a swap in phase k is clause gadget k. After phase k, condition (3) of the invariant only makes a nontrivial claim about the allocation of a variable gadget j for which $L_k = P_j$ and $\mathcal{L}_j = \{P_j, P'_j\}$. In this case, after phase k, the agents matched to the objects n_j, d_j, p'_j, and d'_j are N_j, D_j, P'_j, C''_{m-k+1}, and D'_j, respectively, so the associated claim is satisfied. Moreover, the claims made in condition (3) after phase k that concern other variable gadgets follow from condition (3) before phase k since the only variable gadget involved in any swaps in phase k is variable gadget j.

We now address the "only if" direction. Assume that a sequence S of valid swaps results in agent C''_0 being matched to object c''_m. Thus there are integers $0 = t_0 < t_1 < \cdots < t_m$ such that for $0 \leq k \leq m$, agent C''_0 first becomes matched to object c''_k at "time" t_k, i.e., immediately after the first t_k swaps of S have been performed. For any integer k such that $0 \leq k \leq m$, let

$Q(k)$ denote the predicate "at time t_k, agent C'_i is matched to object c''_{i-1} for all i such that $1 \leq i \leq k$". We now use induction on k to prove that $Q(k)$ holds for $0 \leq k \leq m$. It is easy to see that $Q(0)$ holds. Let k be an integer such that $1 \leq k \leq m$ and assume that $Q(k-1)$ holds. We need to prove that $Q(k)$ holds. Since $Q(k-1)$ holds, each agent in $\{C_i \mid 1 \leq i < k\}$ is matched to its favorite object at time t_{k-1}, and hence does not move thereafter. Thus, to establish that $Q(k)$ holds, it is sufficient to prove that the agent, call it a, moving from object c''_k to object c''_{k-1} in swap t_k of S (which moves agent C''_0 from object c''_{k-1} to object c''_k) is C'_k. Since c''_{k-1} belongs to objects(I, a), we deduce that a belongs to

$$\{C'_i \mid 1 \leq i \leq k\} \cup \{C''_i \mid 0 \leq i < k\}.$$

As discussed above, for $1 \leq i < k$, agent C'_i is permanently matched to object c''_{i-1} as of time t_{k-1}. It follows that a does not belong to $\{C'_i \mid 1 \leq i < k\}$. It also follows that each agent in $\{C''_i \mid 1 \leq i < k\}$ moved up the staircase from object c''_{k-1} to object c''_k prior to time t_{k-1}, and hence can never return to object c''_{k-1}; thus agent a does not belong to $\{C''_i \mid 1 \leq i < k\}$. Since agent a is not equal to C''_0, we conclude that agent a is equal to C'_k, as required. This completes our proof by induction that $Q(k)$ holds for $0 \leq k \leq m$.

Since $Q(m)$ holds, we know that for each clause index i, the sequence of swaps S causes agent C'_i to move away from its initial object c'_i. For any clause index i, let A_i denote $\{a \in A \mid c'_i \in \text{objects}(I, a)\} - C'_i$. We can only swap agent C'_i away from its initial object c'_i in favor of some agent in A_i.

Our reduction ensures that A_i is equal to the set of agents corresponding to literals satisfying clause i, in the following sense: for each negative literal $\neg x_j$ appearing in C_i, the agent N_j belongs to A_i; for each positive literal x_j such that the first (resp., second) occurrence of x_j appears in C_i, the agent P_j (resp., P'_j) belongs to A_i. For each clause index i, let L_i denote the agent in A_i that swaps with agent C'_i when C'_i moves away from its initial object c'_i, and let ℓ_i denote the literal corresponding to L_i. We claim that it is possible to find a truth assignment for f that simultaneously sets all of the literals ℓ_i to true, and thus satisfies f. To prove this, it suffices to show that for any variable index j, if N_j belongs to $\{L_i \mid 1 \leq i \leq m\}$ then $\{L_i \mid 1 \leq i \leq m\} \cap \{P_j, P'_j\} = \emptyset$. Below we prove that the following stronger claim holds: For any variable index j, if a sequence of swaps causes agent N_j to leave variable gadget j (i.e., to move from object p_j to object c''_m) then agents P_j and P'_j remain in variable gadget j under this sequence of swaps.

To prove the latter claim, let us fix a sequence of swaps S that causes agent N_j to leave variable gadget j. The only agent $a \neq N_j$ such that object n_j belongs to objects(I, a) is P'_j. Since N_j moves away from its initial object n_j under S, we deduce that S includes two swaps moving agent P'_j first to object d_j and then to object n_j. Since object n_j is a leaf, agent P'_j remains matched to object n_j thereafter. Since agent N_j does not remain matched to object d_j, it is eventually swapped to object p'_j. Since agent D_j has previously moved from object d_j to object p'_j, it cannot move back to object d_j. The only agent $a \notin \{N_j, D_j, P'_j\}$ such that d_j belongs to objects(I, a) is agent P_j. Thus,

the swap that moves agent N_j from object to d_j to object p'_j moves agent P_j from object p'_j to object d_j. Since agent P'_j is permanently matched to object n_j, we conclude that agent P_j is permanently matched to object d_j. Thus if agent N_j leaves variable gadget j (indeed, if N_j merely reaches object p'_j), then neither agent P_j nor agent P'_j leaves variable gadget j. □

2.6.2 NP-Completeness for Reachable Matching on Cliques

We begin by proving in Lemma 2.6.2 that reachable matching on general graphs is NP-complete. We use Lemma 2.6.2 to establish that reachable matching on cliques is also NP-complete.

Lemma 2.6.2. *Reachable matching on general graphs is NP-complete.*

Proof. It is easy to see that reachable matching on general graphs belongs to NP. We use a reduction from reachable object on generalized stars to reachable matching on general graphs to establish that reachable matching on general graphs is NP-complete.

Fix an arbitrary reachable object on generalized stars instance I. Without loss of generality, we can assume that the associated configuration $\chi = (F, \mu)$ is such that $F = (A, B, \succ, E)$, $A = \{a_1, \ldots, a_n\}$, $B = \{b_1, \ldots, b_n\}$, and $\chi(a_i) = b_i$ for $1 \leq i \leq n$. We can also assume without loss of generality that our goal is to determine whether there is a matching μ_1 in reach(χ) such that $\mu_1(a_1) = b_n$.

Below we describe how to transform reachable object on generalized

stars instance I into a reachable matching on general graphs instance I' such that I is a positive instance of reachable object on generalized stars if and only if I' is a positive instance of reachable matching on general graphs . The reachable matching on general graphs instance I' has two associated configurations $\chi' = (F', \mu')$ and $\chi'' = (F', \mu'')$, where $F' = (A', B', \succ', E')$. The set of agents A' is equal to $A \cup A^*$ where $A^* = \{a_1^*, \ldots, a_n^*\}$. The set of objects B' is equal to $B \cup B^*$ where $B^* = \{b_1^*, \ldots, b_n^*\}$. The perfect matching μ' from A' to B' satisfies $\mu'(a_i) = b_i$ and $\mu'(a_i^*) = b_i^*$ for $1 \leq i \leq n$. The subgraph of (B', E') induced by the set of objects B is equal to (B, E). The subgraph of (B', E') induced by the set of objects B^* is a clique. There are n edges connecting these two subgraphs: there is an edge from object b_i to object b_i^* for $1 \leq i \leq n$. The agent preferences $\succ'$ are defined as follows.

- For any integer i such that $1 \leq i \leq n$, the most preferred object of agent a_i^* is b_i, followed by object b_i^*, followed by the remaining objects in B' in arbitrary order.

- The most preferred object of agent a_1 is b_n^*, followed by the objects in B in the order specified by the preferences of agent a_1 under $\succ$, followed by the objects in $B^* - b_n^*$ in arbitrary order.

- The most preferred objects of agent a_n are $b_1^*, \ldots, b_{n-1}^*$, followed by the objects in B in the order specified by the preferences of agent a_n under $\succ$, followed by object b_n^*.

- For any integer i such that $1 < i < n$, the most preferred objects of a_i are $b_i^*, \ldots, b_{n-1}^*$, followed by $b_{i-1}^*, \ldots, b_1^*$, followed by the objects in B in the order specified by the preferences of agent a_i under $\succ$, followed by object b_n^*.

The perfect matching μ'' associated with configuration χ'' maps each agent in A' to its most preferred object in B'. (Note that μ'' is a perfect matching from A' to B', since no two agents in A' share the same most preferred object.)

It is easy to see that we can construct instance I' in polynomial time in the size of instance I. It remains to argue that instance I is a positive instance of reachable object on generalized stars if and only if I' is a positive instance of reachable matching on general graphs.

We begin by addressing the "only if" direction. Assume that instance I is a positive instance of reachable object on generalized stars. Thus there is a configuration χ_1 in reach(χ) such that $\chi_1(a_1) = b_n$. Our construction of the agent preferences therefore ensures the existence of a configuration χ_1' in reach(χ') such that $\chi_1'(a_i) = \chi_1(a_i)$ and $\chi_1'(a_i^*) = \chi(a_i^*) = b_i^*$ for $1 \leq i \leq n$.

It is easy to check that a swap across edge (b_i, b_i^*) can be applied to configuration χ_1' for $1 \leq i \leq n$. Let χ_2' denote the configuration obtained by applying these n swaps to χ_1'. Thus χ_2' belongs to reach(χ'). Furthermore, it is easy to check that each agent in $A^* + a_1$ is matched in χ_2' to its most preferred object under $\succ'$.

We iteratively construct a sequence of $n-1$ configurations $\chi_3', \ldots, \chi_{n+1}'$

such that configuration χ'_k satisfies the following properties for $3 \leq k \leq n+1$: χ'_k belongs to reach(χ'); every agent in $A \cup \{a_1, \ldots, a_{k-2}\} + a_n$ is matched in χ'_k to its most preferred object in configuration χ'_k. We begin by applying zero or one swaps to configuration χ'_2 to obtain configuration χ'_3. If $\chi'_2(a_n) = b_1^*$, then we define χ'_3 as χ'_2. If not, then $\chi'_2(a_i) = b_1^*$ for some i in $\{2, \ldots, n-1\}$. The preferences of agents a_i and a_n ensure that a swap between these two agents can be applied to configuration χ'_2. We define χ'_3 as the configuration that results from applying this swap. It easy to see that configuration χ'_3 belongs to reach(χ') and that every agent in $A \cup \{a_1, a_n\}$ is matched in χ'_3 to its most preferred object under $\succ'$.

Now fix an integer k such that $4 \leq k \leq n+1$, and inductively assume that we have constructed a configuration χ'_{k-1} in reach(χ') such that every agent in $A \cup \{a_1, \ldots, a_{k-3}\} + a_n$ is matched to its most preferred object under $\succ'$. We apply zero or one swaps to configuration χ'_{k-1} to obtain configuration χ'_k. If $\chi'_{k-1}(a_{k-2}) = b_{k-2}^*$, then we define χ'_k as χ'_{k-1}. If not, then $\chi'_{k-1}(a_i) = b_{k-2}^*$ for some i in $\{k-1, \ldots, n-1\}$. The preferences of agents a_{k-2} and a_i ensure that a swap between these two agents can be applied to configuration χ'_{k-1}. We define χ'_k as the configuration that results from applying this swap. It easy to see that configuration χ'_k belongs to reach(χ') and that every agent in $A \cup \{a_1, \ldots, a_{k-2}\} + a_n$ is matched in χ'_k to its most preferred object under $\succ'$.

Since χ'_{n+1} belongs to reach(χ') and every agent in A' is matched in χ'_{n+1} to its most preferred object under $\succ'$, we conclude that $\chi'_{n+1} = \chi''$ and hence that I' is a positive instance of reachable matching on general graphs.

Now we address the "if" direction. Assume that instance I' is a positive instance of reachable matching on general graphs. Thus χ'' belongs to reach(χ'), and hence there is a sequence of swaps S that transforms configuration χ' into configuration χ''.

By examining the preferences of the agents in A^*, we deduce that each agent in A^* participates in exactly one swap in S, and that the other agent participating in each of these swaps belongs to A. By examining the preferences of the agents in A, we deduce that agent a_1 is the agent that swaps with agent a_n^*, and that once an agent in A becomes matched to an object in B^*, it remains matched to an object in B^* thereafter. It follows that there is a permutation π of $\{1, \ldots, n\}$ such that $\pi(n) = 1$ and a_i^* swaps with $a_{\pi(i)}$ for $1 \leq i \leq n$.

For any integer k such that $0 \leq k \leq |S|$, let χ'_k denote the configuration reached by applying the first k swaps of sequence S to configuration χ'. Thus $\chi' = \chi'_0$, $\chi'' = \chi'_{|S|}$, and χ'_k is of the form (F', μ'_k) where μ'_k is a perfect matching from A' to B'.

For any integer k such that $0 \leq k \leq |S|$, we use the perfect matching μ'_k to construct a perfect matching μ_k from A to B, as follows: for each agent a_i^* in A^* such that $\mu'_k(a_i^*)$ belongs to B^*, we define $\mu_k(a_{\pi(i)})$ as $\mu'_k(a_{\pi(i)})$; for each agent a_i^* in A^* such that $\mu'_k(a_i^*)$ belongs to B, we define $\mu_k(a_{\pi(i)})$ as $\mu'_k(a_i^*)$.

For any integer k such that $0 \leq k \leq |S|$, we define χ_k as the configuration (F, μ_k). It is straightforward to prove by induction on k that χ_k belongs

to reach(χ) for $0 \leq k \leq |S|$.

Let ℓ denote the least integer such that $\chi_\ell^{-1}(b_n) = a_n^*$. We know that ℓ exists since $\chi_{|S|}^{-1}(b_n) = a_n^*$, and that ℓ is positive since $\chi_0^{-1}(b_n) = a_n$. As discussed earlier, agent a_1 is the only agent that participates in a swap with agent a_n^*. Hence $\chi_{\ell-1}^{-1}(b_n) = a_1$. Since $\chi_{\ell-1}$ belongs to reach(χ), we conclude that I is a positive instance of reachable object on generalized stars, as required. $\square$

It is easy to see that the reachable matching on cliques problem belongs to NP. We use a reduction from reachable object on cliques to reachable matching on cliques to establish that reachable matching on cliques is NP-complete. This reduction is similar to the one used in the proof of Lemma 2.6.2.

Fix an arbitrary reachable object on cliques instance I. Without loss of generality, we can assume that the associated configuration $\chi = (F, \mu)$ is such that $F = (A, B, \succ, E)$, $A = \{a_1, \ldots, a_n\}$, $B = \{b_1, \ldots, b_n\}$, and $\chi(a_i) = b_i$ for $1 \leq i \leq n$. We can also assume without loss of generality that our goal is to determine whether there is a matching μ_1 in reach(χ) such that $\mu_1(a_1) = b_n$.

We now describe how to transform reachable object on cliques instance I into a reachable matching on cliques instance I'. Instance I' has two associated configurations $\chi' = (F', \mu')$ and $\chi'' = (F', \mu'')$, where $F' = (A', B', \succ', E')$. The set of agents A' is equal to $A \cup A^*$ where $A^* = \{a_1^*, \ldots, a_n^*\}$. The set of objects B' is equal to $B \cup B^*$ where $B^* = \{b_1^*, \ldots, b_n^*\}$. Let K_{2n} denote the complete graph with vertex set B', and let E' denote the edge set of K_{2n}. The agent preferences $\succ'$ and the matchings μ' and μ'' are as described in the proof

of Lemma 2.6.2.

Let $\hat{E}$ denote the union of three sets of edges: $\{(b_i, b_j) \mid i, j \in [n] \wedge i \neq j\}$; $\{(b_i, b_i^*) \mid i \in [n]\}$; $\{(b_i^*, b_j^*) \mid i, j \in [n] \wedge i \neq j\}$. Lemma 2.6.3 below establishes that if a swap occurs on an edge e in I', then e belongs to $\hat{E}$.

Lemma 2.6.3. *Let i and j be elements of $[n]$ such that $i \neq j$. Let μ_1 and μ_2 be matchings in* reach(χ') *such that $\mu_1 \rightarrow_F \mu_2$. Then $\mu_2^{-1}(b_i) \neq \mu_1^{-1}(b_j^*)$.*

Proof. We consider two cases.

Case 1: $\mu_1^{-1}(b_j^*)$ belongs to A^*. By examining the preferences of agents in A^*, we deduce that $\mu_1^{-1}(b_j^*) = a_j^*$. The only object that agent a_j^* prefers to b_j^* is b_j. Hence $\mu_2^{-1}(b_i) \neq a_j^* = \mu_1^{-1}(b_j^*)$.

Case 2: $\mu_1^{-1}(b_j^*)$ belongs to A. By examining the preferences of agents in A, we deduce that $\mu_2(\mu_1^{-1}(b_j^*))$ belongs to B^*. Hence $\mu_2^{-1}(b_i) \neq \mu_1^{-1}(b_j^*)$. □

Using Lemma 2.6.3, along with the same reasoning as in the proof of Lemma 2.6.2, we deduce that I' is a positive instance of reachable matching on cliques if and only if I is a positive instance of reachable object on cliques. Thus Theorem 2.6.4 below holds.

Theorem 2.6.4. *Reachable matching on cliques is NP-complete.*

2.6.3 NP-Hardness for Pareto-Efficiency on Cliques

Theorem 2.6.5. *Pareto-efficient matching on cliques is NP-hard.*

Proof. We use the same reduction as we used in Section 2.6.2 to establish the NP-completeness of reachable matching on cliques. In analyzing that reduction, we proved that a given instance of reachable object on cliques is positive if and only if every agent gets its most preferred object in the corresponding instance of reachable matching on cliques. Therefore an efficient algorithm for computing a Pareto-efficient matching on cliques yields an efficient algorithm for reachable object on cliques. Since reachable object on cliques is NP-complete, we deduce that Pareto-efficient matching on cliques is NP-hard. □

2.7 Other Polynomial-Time Bounds

In this section, we briefly discuss simple algorithms that serve to justify the remaining polynomial-time entries in Table 2.1.

For reachable matching on trees, the corresponding algorithm of Gourvès et al. [91] for the object-moving model can also be used for the agent-moving model. In particular, for the agent-moving model, Section 2.7.1 shows that reachable matching problem on trees can be solved in $O(n^2)$ time.

For the other two polynomial-time table entries for stars, observe that once an agent swaps away from the center object, it cannot participate in another swap. This observation severely restricts the swap dynamics, making it easy to establish that the reachable object problem on stars can be solved in $O(n^2)$ time and Pareto-efficient matching problem on stars can be solved in $O(n)$ time.

2.7.1 Reachable Matching on Trees

Theorem 2.7.1. *Reachable matching on trees can be solved in $O(n^2)$ time.*

Proof. Let μ and μ' be the initial and target perfect matchings in an instance of the reachable matching problem on a tree. To solve the problem, we use the approach presented by Gourvès et al. [91] for the object-moving model. Observe that every agent a moves along a unique dipath in the tree from $\mu(a)$ to $\mu'(a)$. Therefore, it suffices to check whether there is a sequence of swaps such that each agent moves according to its dipath. Let us say that an edge is good if the two agents currently matched to the endpoints of the edge both need to cross the edge as the next step along their respective dipaths. Using essentially the same argument as in the proof of Proposition 3 in Gourvès et al. [91], we can show if the target matching is reachable and is not equal to the current matching, then at least one good edge exists. It follows easily that if the target matching is reachable, then we can reach it by repeatedly performing a swap across an arbitrary good edge.

We now discuss how to find good edges efficiently so that the overall running time is $O(n^2)$. We begin by traversing the tree to construct a set containing all of the edges that are good in the initial state. The order of the edges within this set is immaterial, so it can be implemented using a simple data structure such as a list or stack. Then, while the set of good edges is nonempty, we perform the following steps. First, we remove an arbitrary good edge from the set, and perform the associated swap. It is easy to see that the

other edges in the good set remain good after the swap. Furthermore, up to two edges can become good as a result of the swap, and a simple local search can be used to identify these edges in constant time. Any newly-identified good edges are added to the set of good edges, and we proceed to the next iteration.

Initialization of the set of good edges takes $O(n)$ time and each iteration of the loop performs one swap and takes constant time. Since the number of swaps is $O(n^2)$, the claimed time bound follows. □

2.7.2 Algorithms for Stars

In this section, we solve the reachable object and Pareto-efficient matching problems on stars. We refer to the object located at the center of the star as the center object, and to the remaining objects as leaf objects. We refer to the agent that is currently matched to the center object as the center agent O, and to the remaining agents as leaf agents.

Theorem 2.7.2. *Reachable object on stars can be solved in $O(n^2)$ time.*

Proof. Let a be the given agent and let b the given target object in an instance of the reachable object problem. Notice that there is a unique dipath for a to follow to reach b, and the dipath has at most two edges.

There are three cases for the dipath, from center to leaf, from leaf to center, and from leaf to leaf. The center to leaf case is straightforward as the

only way for agent a to reach object b is to perform a single swap involving both of them.

For the leaf to center case, we use the approach presented by Gourvès et al. [91] for the object-moving model. The problem reduces to the search of a path in a digraph $G = (A, E)$ where $(a, a') \in E$ with $a \in A, a' \in A \setminus \{O\}$ if and only if a and a' can rationally trade when a is at the center position and a' is in its initial position. There is a path from O to a in G if and only if a can move to the center position. It takes $O(n^2)$ time to construct G and solve this path problem.

We reduce the leaf to leaf case to the leaf to center case, as follows. First, we solve a leaf to center problem to determine whether agent a can be moved to the center without moving the initial owner of object b. If this is possible, then we check whether agent a and the initial owner of object b can trade their objects. This procedure works correctly because the ownership of any leaf object can change at most once in any valid sequence of swaps. To see this, observe that once an agent moves from the center to a leaf, it can never return to the center. Accordingly, for agent a to move to object b, the initial owner of b needs to remain stationary until a reaches the center. □

Theorem 2.7.3. *Pareto-efficient matching on stars can be solved in $O(n)$ time.*

Proof. We use an algorithm based on serial dictatorship [4]. First, we prune any leaf agent with its associated object if the agent does not prefer the center

object to its own object, since this leaf agent will never be involved in any swap. Therefore, the center agent can swap with any remaining leaf agent only if the leaf agent holds one of its preferred objects. If the top preference of the center agent out of the remaining objects is the center object, then the current matching is Pareto-efficient as no swaps can occur. Otherwise, the top preference of the center agent a out of the remaining objects is a leaf object b, and we can apply the swap operation that moves agent a to object b. The previous owner of object b becomes the new center agent. Then, we prune agent a and object b and recurs on the new center agent. It is easy to see that this whole process takes $O(n)$ time, and when each agent is pruned, it holds its favorite object among the remaining objects. It follows that the matching returned by this process is not Pareto-dominated by any other matching, and hence is Pareto-efficient. □

Chapter 3

Maximum Votes Pareto-Efficient Fair Allocations

3.1 Introduction

Model. Matching indivisible objects to agents is a fundamental problem in both computer science and economics. In a seminal work, Shapley and Scarf [154] introduced the notion of a housing market, which corresponds to the special case of one-sided matching in which there are an equal number of agents and objects, each agent is initially endowed with a distinct object, and each agent is required to be matched to exactly one object. Fruitful applications have arisen from the housing market problem: assigning virtual machines to servers in cloud computers, and allocating graduates to trainee positions. There are two different assumptions regarding preference relations of agents. One is strict, which is a full ordinal list of all objects, and the other one is weak, where agents are allowed to be indifferent between objects. Both preference relations have been widely studied.

However, in practice, the assumption that any agent is able to trade with any other agent may be too strong. Agents tend to trade with agents that

The results presented in this chapter are based on joint work with Xiong Zheng [120], and I am the primary contributor for these results.

they know and trust, and it is hard to let two agents who do not trust each other exchange their objects even if they would each benefit. This motivated Gourvès et al. [91] to initiate the line of research on the object-moving model, as discussed in Chapter 2. A pair of agents are allowed to swap objects with each other only if (1) both of them will be better off after the swap and (2) they are directly connected (socially tied) in the network. As discussed in Section 2.1, Gourvès et al. investigate the swap dynamics of the housing market problem in this model by considering the following three computational problems. The first problem, Reachable Object (RO), asks whether there is a sequence of swaps that results in matching a given agent to a given target object. The second problem, Reachable Matching (RM), asks whether there is a sequence of swaps that results in a given target matching. The third problem, Pareto Efficiency (PE), asks how to find a sequence of swaps that results in a Pareto-efficient matching with respect to the set of reachable matchings. Of particular relevance to the present chapter is the last problem. We remark that, like Gourvès et al., our focus is on Pareto-efficient matchings among reachable matchings (to be formally defined in Section 3.2). In particular, we focus on reachable matchings that are not Pareto-dominated by a reachable matching, as opposed to an arbitrary matching. Specifically, we are interested in finding a sequence of swaps that yields a Pareto-efficient matching that maximizes the number of agents who prefer their match to their initial endowment.

Maximum Votes Pareto-Efficient Matching. In housing markets, we typically seek a matching between houses and agents that optimizes some

social objectives. Pareto efficiency has been widely considered in the context of house allocation [3, 5, 154]. In the example of Fig. 3.1, there is a Pareto-efficient matching that improves only two agents, while there is another Pareto-efficient matching that improves everyone.

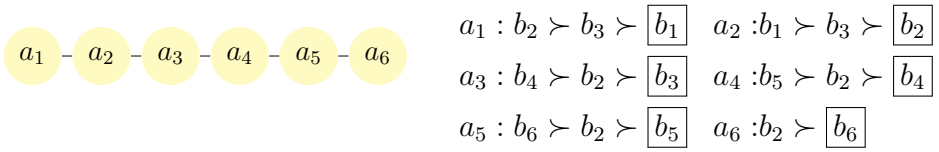


Figure 3.1: Consider an instance with agent set $\{a_1, \ldots, a_6\}$ and object set $\{b_1, \ldots, b_6\}$. The boxed objects represent the initial endowments. Initially, the only possible swaps are between agent pairs (a_1, a_2) and (a_2, a_3). If swap (a_1, a_2) is performed, then we obtain a Pareto-efficient matching in which a_1 and a_2 each gets its best object. If instead swap (a_2, a_3) is performed, then additional swaps between agent pairs (a_1, a_2) and (a_3, a_4) become possible. Indeed, the sequence of swaps (a_2, a_3), (a_1, a_2), (a_3, a_4), (a_4, a_5), (a_5, a_6), yields a different Pareto-efficient matching that improves all agents.

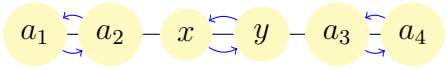
There is a vast literature on refining Pareto-efficiency in object allocation with endowments. Various refining directions are considered, such as fairness ([21, 71]), welfare-maximization ([1, 35]), and stability ([154]). In this work, we propose to refine Pareto-efficiency in the direction of popularity. The notion of popularity was introduced by Abraham et al. [6] for object allocation without endowments. Popularity has gained a lot of attention in the recent past (we refer to the survey by Cseh [61] on this topic). Popular matchings are defined in terms of comparisons between pairs of matchings with respect to the votes of the agents. Consider an election between two matchings M and M' where agents are voters. In this M versus M' election, each agent a

votes for the matching in $\{M, M'\}$ that agent a prefers, i.e., where agent a gets a better assignment. We say that a matching is popular if it never loses a head-to-head election against any matching. However, popular matchings do not always exist: Abraham et al. construct an instance with three agents for which there is no popular matching.

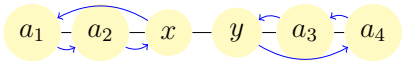
To adapt the popularity notion to housing markets, we introduce the notion of maximum votes matching. We consider the same voting scheme, but focus on the comparison with the initial endowments. Given a reachable matching μ, an agent a votes for the matching if a prefers the object assigned to a by μ to its initial object. We refer to the number of agents that prefer μ to the initial matching as the voting number of μ. We define a maximum votes matching as a reachable matching with the maximum voting number. We remark that a maximum votes matching always exists.

Clearly, not all Pareto efficient matchings have the same voting number. Also, a maximum votes matching is not necessarily a Pareto-efficient matching. Consider the example shown in Fig. 3.2. Part (a) shows a sequence of swaps that gives a Pareto-efficient matching improving all agents. Part (b) shows a sequence of swaps yielding a matching that improves all agents but that is not Pareto-efficient. Thus, it is natural to consider the problem of finding a Pareto-efficient matching with the maximum voting number. We refer to this problem as the maximum votes Pareto-efficient (MVPE) matching problem. In particular, the set of MVPE matchings is the intersection of the following two sets: (1) the set of matchings that are Pareto efficient among

reachable matchings; (2) the set of reachable matchings with the maximum voting number.



(a) A sequence of swaps that results in each agent getting its favorite object.



(b) A sequence of swaps that yields a non-Pareto-efficient matching improving all agents. Agent a_2 obtains object b_x by swapping with agent x. Then agent a_1 gets object b_x by swapping with a_2. Similarly, agent a_4 gets object b_y by the following two swaps: $(y, a_3), (a_3, a_4)$. Clearly, this matching is Pareto-dominated by the matching obtained in part (a).

Figure 3.2: An example in which a non-Pareto-efficient matching improves the most agents. Consider an instance with agent set $\{a_1, a_2, x, y, a_3, a_4\}$ and object set $\{b_1, b_2, b_x, b_y, b_3, b_4\}$. The preferences of the agents and the initial endowments (represented as boxed objects) are given below. The arrows in the figures show where the initial object of each agent goes.

$a_1 : b_2 \succ b_x \succ \boxed{b_1}$ $\quad$ $x : b_y \succ b_2 \succ \boxed{b_x}$ $\quad$ $a_3 : b_4 \succ b_y \succ \boxed{b_3}$

$a_2 : b_1 \succ b_x \succ \boxed{b_2}$ $\quad$ $y : b_x \succ b_3 \succ \boxed{b_y}$ $\quad$ $a_4 : b_3 \succ b_y \succ \boxed{b_4}$

Related work. There are several works on ensuring popularity. For settings where a popular matching does not exist, Kavitha et al. [105] study how to minimally augment the preferences to guarantee the existence of a popular matching. This problem has been shown to be NP-hard. Another way to ensure popularity is to consider mixed matchings, i.e., lotteries over matchings; the popularity property is straightforward to carry over to this setting. Kavitha et al. [104] show that a popular mixed matching always exists and propose an efficient algorithm to find one. McCutchen [127] proposes a

least-unpopularity criterion to find the "most" popular matching; finding this least-unpopular matching is NP-hard.

Our results. Our main contributions are summarized in Table 3.1. Our main positive result is a polynomial-time algorithm for finding a maximum votes Pareto-efficient matching in a path network with strict preferences. To achieve this result, we first present an algorithm for finding a maximum votes matching by studying the structure of a reachable matching in a path network. Specifically, we show that the improved agents in any reachable matching can be partitioned into a set of agent intervals. Each of these agent intervals satisfies the following conditions: the agent interval is a set of consecutive agents on the path; there is a sequence of swaps within the interval that improves all of the agents in the interval. Moreover, given a partition of such agent intervals, we can recover a sequence of swaps improving all of the agents in this collection. Thus, we reduce the problem of computing the maximum voting number to the problem of finding such a partition that includes as many agents as possible; polynomial-time algorithms are known for the latter problem.

To find a maximum votes Pareto-efficient matching, observe that there is a maximum votes matching that is also Pareto-efficient. Thus, to find a maximum votes Pareto-efficient matching, it is enough to find a Pareto-efficient matching among maximum votes matchings. To achieve that, we carefully adapt the serial dictatorship algorithm [4] to ensure that each time a dictator agent a is assigned to an object b, the current partial assignment remains con-

Table 3.1: Summary of results and related work for MVPE matching problem in the object-moving models. The results in parentheses are implied by the hardness results in Table 2.1 or other table entries.

	Strict	Weak
Path	poly-time [Section 3.3]	NP-hard ([95])
Star	poly-time [Section 3.4]	NP-hard [Section 3.4]
Generalized Star	NP-hard [Section 3.5]	(NP-hard)
Tree	(NP-hard)	(NP-hard)
Clique	(NP-hard)	(NP-hard)

sistent with a maximum votes matching. In particular, we present a subroutine to find the most preferred object b of agent a such that there is a sequence of swaps that improves as many agents as possible and results in object b being assigned to agent a. In Section 3.3.3, we use a delicate induction argument to prove the correctness of this subroutine.

In Section 3.4, we present a simple polynomial-time algorithm for finding a maximum votes Pareto-efficient matching on stars with strict preferences.

In terms of negative results, in Section 3.4, we use a novel reduction from the 3-SAT problem to prove that the MVPE problem in star networks with weak preferences is NP-hard. In Section 3.5, we prove that the MVPE problem in generalized stars with strict preferences is NP-hard by reducing from the RO problem in generalized stars with strict preferences, which is known to be NP-complete [91]. The NP-hardness of the MVPE problem on cliques with strict preferences follows directly from the hardness result for the Pareto Efficiency problem on cliques included in Table 2.1.

3.2 Preliminaries

We define an object allocation framework (OAF) on a social network as a 4-tuple $F = (A, B, \succcurlyeq, E)$ where A is a set of agents, B is a set of objects such that $|A| = |B|$, $\succcurlyeq$ is a collection of linear orderings $\{\succcurlyeq_a\}_{a \in A}$ over B such that $\succcurlyeq_a$ specifies the preferences (including ties) of agent a over B, and E is the edge set of a social network between agents in A, i.e., the social network is the undirected graph (A, E). When we consider strict preferences, we use the symbol $\succ$.

We define a matching μ of a given OAF $F = (A, B, \succcurlyeq, E)$ as a subset of $A \times B$ such that no agent or object belongs to more than one pair in μ. (Put differently, μ is a matching in the complete bipartite graph of agents and objects.) We say that such a matching is perfect if $|\mu| = |A|$. For any matching μ, we define agents(μ) (resp., objects(μ)) as the set of all matched agents (resp., objects) with respect to μ. For any matching μ and any agent a that is matched in μ, we use the shorthand notation $\mu(a)$ to refer to the object matched to agent a. For any matching μ and any object b that is matched in μ, we use the notation $\mu^{-1}(b)$ to refer to the agent matched to object b.

For any OAF $F = (A, B, \succcurlyeq, E)$, any perfect matching μ of F, and any edge $e = (a, a')$ in E, we define the matching that results from an exchange operation on edge e as $\mu' = \mu \setminus \{(a, \mu(a)), (a', \mu(a'))\} \cup \{(a, \mu(a')), (a', \mu(a))\}$. We say the exchange is rational if $\mu(a') \succcurlyeq_a \mu(a)$ and $\mu(a) \succcurlyeq_{a'} \mu(a')$, denoted as $\mu \rightarrow_{F,e} \mu'$. We call a rational exchange a *swap*. We use $\mu \rightarrow_F \mu'$ to denote that $\mu \rightarrow_{F,e} \mu'$ for some edge e. We write $\mu \rightsquigarrow_F \mu'$ if there exists a matching history

$\mu = \mu_0, \ldots, \mu_k = \mu'$ of matchings of F such that $\mu_{i-1} \rightarrow_F \mu_i$ for $1 \leq i \leq k$.

We define a configuration as a pair $\chi = (F, \mu)$ where F is an OAF and μ is a perfect matching of F. We say that χ is a path, star, or tree configuration if the social network in F is a path, star, or tree. For any configuration $\chi = (F, \mu)$, we define reach(χ) as the set of all perfect matchings μ' of F such that $\mu \rightsquigarrow_F \mu'$.

A matching μ Pareto-dominates a matching μ' if $\mu(a) \succcurlyeq_a \mu'(a)$ for all agents a in A, and there is at least one agent b in B such that $\mu(b) \succ_b \mu'(b)$. A matching is Pareto-efficient if it is not Pareto-dominated by another matching. Throughout this chapter, we restrict the definition of Pareto-efficiency to the set of reachable matchings: A matching μ is Pareto-efficient with respect to a configuration χ if μ belongs to reach(χ) and μ is not Pareto-dominated by another matching in reach(χ).

Maximum Votes Pareto-Efficient Matching Problem: Given a configuration $\chi = (F, \mu)$ and a matching μ', we use $\text{votes}_F(\mu', \mu)$ to denote the number of agents who prefer μ' to μ. We say $\text{votes}_F(\mu', \mu)$ is the voting number of matching μ'. The maximum votes Pareto-efficient matching problem is equivalent to the following optimization problem:

$$\underset{\tau \in \text{ reach}(\chi) \ \wedge \ \tau \text{ is Pareto-efficient}}{\text{argmax}} \text{votes}_F(\tau, \mu).$$

For a given configuration χ, we use mv(χ) to denote the maximum voting number of a matching in reach(χ). We refer to mv(χ) as the voting number of χ.

3.3 Path Networks

In this section, we study the maximum votes Pareto-efficient matching problem in a path configuration. We begin by introducing some notations. For any non-negative integer n, we define $[n]$ as $\{1, \ldots, n\}$. We use $[i, j]$ to denote the set of agents between agent i and agent j (inclusive). Without loss of generality, in this subsection we restrict our attention to OAFs of the form $F = ([i, j], [i, j], \succ, \{(b, b+1) \mid i \leq b < j\})$ for some positive integers i and j such that $i < j$. We write $\succ$ here because we consider only strict preferences in this section. Let $\Pi = \mu_0, \ldots, \mu_k$ denote a matching history where $\mu_i \rightarrow_F \mu_{i+1}$ for $0 \leq i \leq k-1$, i.e., μ_{i+1} is obtained from μ_i by a swap. We have the following observations.

Observation 3.3.1. *Each object only moves in one direction or stays at its initial position in* Π.

We say an object is right-moving (left-moving) in Π if it moves to the right (left). The following observation says that while implementing the matching history Π, the relative location of any two objects moving in the same direction on the path does not change, i.e., if an object b is located to the left (right) of another object b', then object b will always be to the left (right) of object b'.

Observation 3.3.2. *Let* $\Pi = \mu_0, \ldots, \mu_k$ *denote a matching history. For any two objects* b *and* b' *where* $\mu_0^{-1}(b) < \mu_0^{-1}(b')$ *that move along the same direction in* Π, *we have* $\mu^{-1}(b) < \mu^{-1}(b')$ *for all* μ *in* Π.

Let $\kappa(j,k)$ denote the canonical sequence of exchanges that assigns object j to agent k by directly moving it along the path from j to k. Formally, if $j < k$, $\kappa(j,k) = (j,j+1),\ldots,(k-1,k)$. If $j \geq k$, $\kappa(j,k) = (j,j-1),\ldots,(k+1,k)$. For any sequence of exchanges Π , we say that rational(Π) holds if all its exchanges are rational. We remark that $\kappa(j,k)$ was defined, and some associated properties were proved, by Gourvès et al. [91]. The following two lemmas present some useful structural properties.

Lemma 3.3.3. *Let F be a path OAF, let a be a leaf agent in χ, let a' be an agent in χ such that $a' \neq a$, and let μ_0 and ν be the two matchings such that $\mu_0 \rightsquigarrow_F \nu$ holds and $a = \nu^{-1}(\mu_0(a'))$. Let μ_1 denote the matching obtained from μ_0 by applying an exchange on edge $(a', a'-1)$. Then both $\mu_0 \rightsquigarrow_F \mu_1$ and $\mu_1 \rightsquigarrow_F \nu$ hold.*

Proof. Without loss of generality, we assume that $a < a'$. Hence agent a is the leftmost leaf agent. Let $\Pi = \mu_0,\ldots,\mu_l$ denote a matching history such that $\mu_l = \nu$. We deduce by $a = \nu^{-1}(\mu_0(a'))$ that $\mu_0(a')$ is left-moving in Π and hence there is a matching μ_i in Π such that $\mu_i(a'-1) = \mu_0(a')$. Therefore, agent $a'-1$ prefers $\mu_0(a')$ to its initial endowment $\mu_0(a'-1)$. To prove that $\mu_0 \rightsquigarrow_F \mu_1$ holds, it remains to prove that $\mu_0(a'-1) \succ_{a'} \mu_0(a')$. Since $\mu_0(a')$ moves to the leftmost agent a, we deduce by Observation 3.3.2 that $\mu_0(a'-1)$ is not left-moving or stationary. Therefore, object $\mu_0(a'-1)$ is right-moving and hence there is a matching μ_j in Π such that $\mu_j(a') = \mu_0(a'-1)$. Therefore, agent a' prefers $\mu_0(a'-1)$ to the initial endowment $\mu'_0(a')$, i.e., $\mu_0(a'-1) \succ_{a'} \mu_0(a')$.

We now prove that $\mu_1 \rightsquigarrow_F \nu$ holds. Let $\mathcal{E} = (e_1, \ldots, e_l)$ denote the ordered list of edges such that $\mu_{i-1} \rightarrow_{F,e_i} \mu_i$ holds for all i in $[\ell]$. Let $\mathcal{E}[i,j]$ denote the sub-list of $\mathcal{E}$ indexed from i to j with indices $1 \leq i < j \leq \ell$. Let e_t be the first edge such that $e_t = (a'-1, a')$, i.e., $\mu_{t-1}(a') = \mu_0(a')$, $\mu_{t-1}(a'-1) = \mu_0(a'-1)$, $\mu_t(a'-1) = \mu_0(a')$, and $\mu_t(a') = \mu_0(a'-1)$. Therefore, any edge preceding e_t in $\mathcal{E}$ does not include agents $a'-1$ or a' and does not involve objects $\mu'_0(a'-1)$ or $\mu'_0(a')$. It is straightforward to prove that if there is an index i in $\{2, \ldots, \ell\}$ such that e_{i-1} and e_i are disjoint, then switching the order of the swaps performed on e_{i-1} and e_i in Π yields another matching history Π' that transforms μ_0 to ν. Therefore, by exchanging the swaps performed on e_t with the swaps performed on edges $e_{t-1}, \ldots, e_1$ one by one, we obtain a matching history Π'' that starts with the swap on e_t and transforms μ_0 to ν. We remark that the matching μ_1 is the matching obtained by applying a swap on e_t from μ_0. Therefore, the matching history Π'' yields a matching history that transforms μ_1 to ν, i.e., $\mu_1 \rightsquigarrow_F \nu$ holds. □

Lemma 3.3.4. *Let $\chi = (F, \mu)$ be a path configuration, let a be a leaf agent in χ, let a' be an agent in χ, and let ν be a matching in* reach(χ) *such that $a = \nu^{-1}(\mu(a'))$. Then* rational$(\kappa(a', a))$ *holds and ν belongs to* reach$((F, \mu'))$*, where μ' is the matching that results by applying $\kappa(a', a)$ to μ.*

Proof. Let μ_0 denote μ. Therefore, by recursively applying Lemma 3.3.3 to matchings μ_{i-1} and ν such that $\mu_{i-1} \rightsquigarrow_F \nu$ holds and $\nu(a) = \mu_{i-1}(a'-i+1)$ for each $i = 1, \ldots, a'-a$, we obtain a matching μ_i such that $\mu_{i-1} \rightarrow_{F,(a'-i+1,a'-i)} \mu_i$

and $\mu_i \rightsquigarrow_F \nu$ hold. Thus rational $\kappa(a', a)$ holds. Furthermore, we have that $\mu_{a'-a} = \mu'$ and $\mu_{a'-a} \rightsquigarrow_F \nu$. Hence ν belongs to reach$((F, \mu'))$. □

3.3.1 Maximum Votes Pareto-Efficient Matching

In this section, we present a polynomial-time algorithm for finding a maximum votes Pareto-efficient matching of a given path configuration. In order to find an MVPE matching, we first present an algorithm for computing the maximum number of a given configuration $\chi = (F, \mu)$, i.e., for computing $\text{mv}(\chi)$. We then compute an MVPE matching by combining this algorithm with the serial dictatorship algorithm [4]. In the serial dictatorship algorithm, we let each successive agent (in an arbitrary sequential order) select their most preferred object that has yet to be selected. In our algorithm, agents pick objects according to the order of their IDs, i.e., the smallest agent picks first and the largest agent picks last. When agent a selects an object, we require a to select its most preferred object b such that there exists a matching μ' in reach(χ) with $\mu'(a) = b$ and $\text{votes}_F(\mu', \mu) = \text{mv}(\chi)$. The correctness of our approach is established in Theorem 3.3.18 below.

3.3.2 Maximum Number of Votes

In this section we present and analyze Algorithm 2, which computes the maximum voting number.

Throughout this section, let n denote a fixed positive integer, and let

Algorithm 2: MV(χ)

Input: A path configuration $\chi = (F, \mu)$
Output: The maximum voting number of χ
$\mathcal{I} \leftarrow \emptyset$
for $i \to 1$ **to** $n - 1$ **do**
  for $j \to i + 1$ **to** n **do**
    if $\exists k \in [i, j]$ *s.t.* rational$(\kappa(k, j)) \wedge$ rational$(\kappa(k + 1, i))$ **then**
      $\mathcal{I} \leftarrow \mathcal{I} \cup \{[i, j]\}$
    end
  end
end
Return the maximum coverage of any disjoint set of intervals in $\mathcal{I}$

$\chi = (F, \mu)$ denote a fixed path configuration where

$$F = ([n], [n], \succ, \{(b, b + 1) \mid 1 \leq b < n\}).$$

Definition 3.3.5 (Directional sequence)**.** *A directional sequence is a sequence of R, L, or S symbols. For any agent i, the ith symbol represents the final moving state of its initial object. Symbol L (resp., R) indicates that it moves to the left (resp., right). Symbol S indicates that it does not move.*

For any matching ν in reach(χ), we define an associated directional sequence as follows. For agent i, if its initial object $\mu(i)$ moves to the left (resp., right) in ν, then the ith symbol in the directional sequence is L (resp., R). If it is stationary, then the ith symbol is S. We use $DS(\nu)$ to denote the directional sequence of ν and $DS(\nu([i, j]))$ to denote the subsequence $DS(\nu)[i, j]$ with respect to agents in $[i, j]$ only.

Definition 3.3.6 (RL-block)**.** *We define an RL-block as a directional sequence consisting of one or more R's followed by one or more L's.*

Any agent with symbol S in $DS(\nu)$ does not get a better allocation in ν; any agent with symbol R or L gets improved by ν due to the definition of swaps coupled with strict preferences.

Observation 3.3.7. *Given a matching ν in* reach(χ)*, $DS(\nu)$ can be uniquely partitioned into a disjoint union of RL-blocks and S symbols.*

For any matching ν in reach(χ), let RLB(ν) denote the set of all intervals $[i, j]$ in $[n]$ such that $DS(\nu)[i, j]$ is an RL-block in the unique partition of $DS(\nu)$ of Observation 3.3.7.

Definition 3.3.8 (RL-interval)**.** *We say that an interval $[i, j]$ is an RL-interval if at least one of the following condition holds: (1)* rational($\kappa(j, i)$)*; (2)* rational($\kappa(i, j)$)*; (3) there exists an integer k in (i, j) such that* rational($\kappa(k + 1, i)$) *and* rational($\kappa(k, j)$)*.*

From the above definition, we can obtain the following observation.

Observation 3.3.9. *For any RL-interval $[i, j]$, there exists a matching history between agents in $[i, j]$ that improves all agents in the interval.*

Lemma 3.3.10. *Let ν be a matching in* reach(χ) *and let $[i, j]$ be in* RLB(ν)*. Then interval $[i, j]$ is an RL-interval.*

Proof. Let $R^m L^{j-i+1-m}$ denote the RL-block $DS(\nu)[i,j]$. Let b_1 and b_2 denote the initial endowment of agent $i+m-1$ and agent $i+m$ in χ, respectively. Thus, b_1 is the rightmost right-moving object and b_2 is the leftmost left-moving object. Therefore, from Observations 3.3.1 and 3.3.2, we have $\nu^{-1}(b_1) = j$ and $\nu^{-1}(b_2) = i$; otherwise, the initial endowments of agent i and agent j are stationary. Hence we deduce that a reachable matching ν maps b_1 to leaf agent j and maps b_2 to leaf agent i. By Lemma 3.3.4, $\kappa(i+m-1, j)$ and $\kappa(i+m, i)$ are both rational. □

We say that the set of intervals is disjoint if they are pairwise disjoint.

Lemma 3.3.11. *For any matching ν in* reach(χ)*, the set of intervals* RLB(ν) *is a disjoint set of RL-intervals.*

Proof. Since each interval in RLB(ν) is an RL-interval (Lemma 3.3.10), each interval in RLB(ν) is an RL-interval. Also, all intervals in RLB(ν) are clearly disjoint. □

Lemma 3.3.12. *For any disjoint set of RL-intervals D, there exists a matching ν in* reach(χ) *such that* RLB$(\nu) = D$.

Proof. We explicitly construct such a matching as follows. For each RL-interval $[i,j]$ in D, from Observation 3.3.9, we know there exists a matching history that improves all agents in $[i,j]$. Thus, matching ν can be constructed by implementing such a matching history for each interval $[i,j]$ in D on the initial matching μ. □

Let $\mathcal{I}$ denote the set of all RL-intervals in χ. We define the coverage of a collection of disjoint intervals in $\mathcal{I}$ as the size of their union. Let MCDI($\mathcal{I}$) denote the maximum coverage of any disjoint set of intervals in $\mathcal{I}$.

Lemma 3.3.13. MCDI($\mathcal{I}$) = mv(χ).

Proof. Lemma 3.3.11 implies that mv(χ) $\leq$ MCDI($\mathcal{I}$). Assume for the sake of contradiction that mv(χ) $<$ MCDI($\mathcal{I}$). Then there exists a disjoint set of RL-intervals such that mv(χ) $< \sum_{[i,j]\in D} j-i+1$. By Lemma 3.3.12, there exists a matching ν in reach(χ) such that RLB(ν) $= D$. For matching ν, we have $\text{votes}_F(\nu,\mu) = \sum_{[i,j]\in \text{RLB}(\nu)} j-i+1 = \sum_{[i,j]\in D} j-i+1 > \text{mv}(\chi)$, a contradiction. $\square$

We remark that Algorithm 2 can be generalized to obtain a polynomial-time algorithm for computing a maximum votes matching of weighted agents. Formally, let $W : [n] \to \mathbb{R}$ denote a weight function assigning each agent a in $[n]$ a real value $W(i)$. For any matchings μ', μ of F, let $\text{votes}_F(\mu', \mu, W) = \sum_{a\in[n]:\mu'(a)\succ_a\mu(a)} W(a)$, i.e., the total weight of the agents that prefer μ' to μ. In addition, let $\text{mv}(\chi, W) = \max_{\mu'\in\text{reach}(\chi)} \text{votes}_W(\mu', \mu)$. To allow for the weight function W, we reduce the problem of computing mv(χ, W) to computing the maximum weighted coverage of any disjoint intervals in $\mathcal{I}$, where each interval $[i, j]$ in $\mathcal{I}$ has weight $\sum_{k\in[i,j]} W(k)$. To compute the maximum weighted coverage of disjoint intervals in $\mathcal{I}$, we construct a directed graph $G = (\mathcal{I}, E)$ as follows. Each node in G is an interval $[i, j]$ in $\mathcal{I}$. For every two nodes $[i, j]$ and $[i', j']$, there is a directed edge from node $[i, j]$ to node $[i', j']$ if $j < i'$. Moreover, each node has a cost equal to the weight of the corresponding interval. Note

that this graph is acyclic, and it takes $O(n^2)$ time to find a maximum-weight path in an acyclic graph.

3.3.3 An MVPE Algorithm

In this section, we present Algorithm 4 for finding an MVPE matching given a path configuration. Our main idea is to apply a serial dictatorship procedure. To simplify the presentation, we focus on the leftmost agent a in any path configuration $\chi = (F, \mu)$ on agents $[a, n]$. We seek the best possible object b for the leftmost agent a such that there exists a maximum votes matching τ in reach(χ) with $\tau(a) = b$. Note that Lemma 3.3.4 implies that rational($\kappa(\mu^{-1}(b), a)$) holds. To match agent a with object b, we apply $\kappa(\mu^{-1}(b), a)$ in χ. Let χ' denote the new configuration on agents $[a + 1, n]$ obtained by truncating the leftmost agent a along with its assigned object b. We then recurse on the leftmost agent $a + 1$ in χ'.

However, when we recurse on χ', the situation is a bit different than in χ. The applied sequence of swaps $\kappa(\mu^{-1}(b), a)$ has improved agents between agent a and $\mu^{-1}(b)$ in χ, and also in χ'. These agents will vote when this serial dictatorship procedure ends. Thus, when we recurse on χ', in order to ensure that the resulting matching is a maximum vote matching, we need to find a matching maximizing votes among agents to the right of $\mu^{-1}(b)$ in χ', rather than all agents.

We now provide a more formal presentation of our results. We begin with some useful notations. For any agent interval $[i, j]$ that is a subset of

$[n]$, let $\chi[i,j]$ and $\mu[i,j]$ denote the truncated configuration and truncated matching induced by agents in $[i,j]$, respectively. Let $\text{mv}(\chi,[i,j])$ denote the maximum number of agents in $[i,j]$ that can be improved by any reachable matching in $\text{reach}(\chi)$, let $\text{votes}_F(\mu',\mu,[i,j])$ denote the number of agents in $[i,j]$ that prefer μ' to μ, and let $\text{MVM}(\chi,[i,j])$ denote the set of reachable matchings with voting number $\text{mv}(\chi,[i,j])$.

We are now ready to present our recursive subroutine called BOLA (best object for the leftmost agent), shown as Algorithm 3. Given a configuration $\chi = (F,\mu)$ on agents $[a,n]$ and an agent $a' \geq a$, Algorithm 3 finds the leftmost agent a's most preferred object b^* such that there exists a μ' in $\text{MVM}(\chi,[a',n])$ with $\mu'(a) = b^*$. In Algorithm 3, agent a' is used to limit the objects that agent a can be possibly matched with. Specifically, Algorithm 3 only considers objects that are initially owned by agents in $[a',n]$. In Algorithm 2, we need to compute $\text{mv}(\chi,[a,n])$ for a given χ on the agents in $[a,n]$. This can be done by applying the weighted version of Algorithm 2 as discussed in Section 3.3.2.

Lemma 3.3.14 below establishes a one-to-one map between the set of matchings in $\text{reach}(\chi)$ that match agent a to object $\mu(i)$ and the set of matchings in $\text{reach}(\chi'[a+1,n])$.

Lemma 3.3.14. *Let $\chi = (F,\mu)$ be a configuration with agents $[a,n]$, let i be an agent in $[a,n]$ such that $\text{rational}(\kappa(i,a))$ holds, let μ' be the matching that results from applying $\kappa(i,a)$ to μ, let $\chi' = (F,\mu')$, and let ν be a matching of*

Algorithm 3: BOLA(χ, a')

Input: A configuration $\chi = (F, \mu)$ with agents $[a, n]$, an agent $a' \in [a, n]$

Output: The most preferred object b^* of the leftmost agent, where there is a matching μ' in $\mathrm{MVM}(\chi, [a', n])$ assigning b^* to a

```
// Remark: a is the leftmost agent in χ
b*, V ← μ(a), mv(χ, [a', n])
for i → a to n do
    if μ(i) ≽_a b* ∧ rational(κ(i, a)) then
        // Remark: i denotes an agent
        μ* ← the matching that results from applying κ(i, a) to μ
        χ* ← (F, μ*)
        a'' ← max(i + 1, a')
        if votes(μ*, μ, [a', a'' − 1]) + mv(χ*, [a'', n]) = V then
            b* ← μ(i)
        end
    end
end
return b*
```

F. Then ν belongs to $\mathrm{reach}(\chi, \{(a, \mu(i))\})$ *if and only if $\nu[a+1, n]$ belongs to* $\mathrm{reach}(\chi'[a+1, n])$.

Proof. The only if direction follows by Lemma 3.3.4. For the if direction, observe that by concatenating $\kappa(i, a)$ and the matching history that transforms $\mu'[a+1, n]$ to $\nu[a+1, n]$, we get a matching history that transforms μ to ν. Henceν belongs to reach(χ). □

Lemma 3.3.15 below establishes some basic results that are useful for proving Lemma 3.3.16 and Theorem 3.3.18.

Lemma 3.3.15. *Let $\chi = (F, \mu)$ be a configuration with agents $[a, n]$, let a' be an agent in $[a, n]$, let b be an object, let $a'' = \max(\mu^{-1}(b) + 1, a')$, let τ be a matching in* reach(χ) *such that $\tau(a) = b$ and let μ' be the matching that results from applying $\kappa(\mu^{-1}(b), a)$ to μ. Then the value* votes$(\tau, \mu, [a', n])$ *is equal to the sum of* votes$(\mu', \mu, [a', a'' - 1])$ *and* votes$(\tau[a + 1, n], \mu'[a + 1, n], [a'', n])$. *Next, if ν is a matching in MVM$(\chi, [a', n])$ such that $\nu(a) = b$, then $\nu[a+1, n]$ belongs to MVM$(\chi'[a + 1, n], [a'', n])$ where $\chi' = (F, \mu')$.*

Proof. Note that votes$(\tau, \mu, [a', n])$ = votes$(\tau, \mu, [a', a''-1])$+votes$(\tau, \mu, [a'', n])$. To prove the desired equation, it suffices to prove that votes$(\tau, \mu, [a', a'' - 1])$ = votes$(\mu', \mu, [a', a'' - 1])$ and votes$(\tau, \mu, [a'', n])$ = votes$(\tau[a + 1, n], \mu'[a + 1, n], [a'', n])$. First, we prove that

$$\text{votes}(\tau, \mu, [a', a'' - 1]) = \text{votes}(\mu', \mu[a', a'' - 1]).$$

We deduce by Lemma 3.3.4 that τ belongs to reach(χ'), and hence all agents improved by μ' are improved by τ, i.e.,

$$\text{votes}(\tau, \mu, [a', a'' - 1]) \geq \text{votes}(\mu', \mu, [a', a'' - 1]).$$

Observe that $\kappa(\mu^{-1}(b), a)$ improves all agents in $[a', a'' - 1]$. Therefore, there are not any matchings that improve more than votes$(\mu', \mu, [a', a''-1])$ agents in $[a', a''-1]$. That is, votes$(\tau, \mu, [a', a''-1]) \leq$ votes$(\mu', \mu, [a', a''-1])$. Therefore, votes$(\tau, \mu, [a', a'' - 1])$ = votes$(\mu', \mu, [a', a'' - 1])$. Second, we show that

$$\text{votes}(\tau, \mu, [a'', n]) = \text{votes}(\tau[a + 1, n], \mu'[a + 1, n], [a'', n]).$$

We start with the observation that $a'' = \max(\mu^{-1}(b) + 1, a') \geq a + 1$. Thus $\text{votes}(\tau, \mu, [a'', n]) = \text{votes}(\tau[a + 1, n], \mu[a + 1, n], [a'', n])$. Note that $\mu[a'', n] = \mu'[a'', n]$ as $\kappa(\mu^{-1}(b), a)$ does not contain any agents in $[a'', n]$. Therefore, $\text{votes}(\tau[a + 1, n], \mu[a + 1, n], [a'', n]) = \text{votes}(\tau[a + 1, n], \mu'[a + 1, n], [a'', n])$, which yields $\text{votes}(\tau, \mu, [a'', n]) = \text{votes}(\tau[a + 1, n], \mu'[a + 1, n], [a'', n])$.

It remains to prove the second claim of the lemma. We deduce by Lemma 3.3.14 that $\tau[a + 1, n]$ belongs to $\text{reach}(\chi'[a + 1, n])$. Assume for the sake of contradiction that $\text{votes}(\tau[a + 1, n], \mu'[a + 1, n], [a'', n]) < \text{mv}(\chi'[a + 1, n], [a'', n])$. That is, there is a matching τ' such that $\tau'(a) = b$, $\tau'[a + 1, n]$ belongs to $\text{reach}(\chi'[a + 1, n])$, and $\text{votes}(\tau'[a + 1, n], \mu'[a + 1, n], [a'', n]) > \text{votes}(\tau[a + 1, n], \mu'[a + 1, n], [a'', n])$. We deduce by Lemma 3.3.14 that τ' belongs to $\text{reach}(\chi)$. Since τ' is a matching in $\text{reach}(\chi)$ such that $\tau'(a) = b$, we can apply the first part of the lemma (with τ' playing the role of τ), to deduce that $\text{votes}(\tau', \mu, [a', n]) = \text{votes}(\mu', \mu, [a', a'' - 1]) + \text{votes}(\tau'[a + 1, n], \mu'[a + 1, n], [a'', n])$. Thus

$$\text{votes}(\tau', \mu, [a', n]) > \text{votes}(\tau, \mu, [a', n]),$$

a contradiction since τ belongs to $\text{MVM}(\chi, [a', n])$. □

Lemma 3.3.16 characterizes all possible maximum votes matchings in terms of the two **if** conditions used in Algorithm 3. Next we use Lemma 3.3.16 to prove Lemma 3.3.17, which is our main correctness lemma. It implies that to find the leftmost agent a's most preferred object b^* such that there exists a matching μ' in $\text{MVM}(\chi, [a', n])$ with $\mu'(a) = b^*$, we can iterate through all

objects that satisfy the two **if** conditions and find the one that is preferred by agent a.

Lemma 3.3.16. *Let $\chi = (F, \mu)$ denote an configuration, let $[a, n]$ denote the set of agents in χ, let a' denote an agent in $[a, n]$, let τ denote a matching in* reach(χ)*, let $b = \tau(a)$, and let $a'' = \max(\mu^{-1}(b) + 1, a')$. Then the matching τ belongs to MVM$(\chi, [a', n])$ if and only if the following two conditions hold:*

(1) rational$(\kappa(\mu^{-1}(b), a))$*;*

(2) votes$(\mu', \mu, [a', a'' - 1]) +$ mv$(\chi'[a + 1, n], [a'', n]) =$ mv$(\chi, [a', n])$*, where μ' is obtained from applying $\kappa(\mu^{-1}(b), a)$ to μ and $\chi' = (F, \mu')$.*

Proof. We begin with the only if direction. Lemma 3.3.4 implies condition (1) since τ belongs to MVM$(\chi, [a', n])$. To prove condition (2), since τ belongs to MVM$(\chi, [a', n])$, we deduce by Lemma 3.3.15 that $\tau[a + 1, n]$ belongs to MVM$(\chi'[a+1, n], [a'', n])$, i.e., votes$(\tau[a+1, n], \mu'[a+1, n], [a'', n]) =$ mv$(\chi'[a+1, n], [a'', n])$. Furthermore, by Lemma 3.3.15, we deduce that votes$(\tau, \mu, [a', n]) =$ votes$(\mu', \mu, [a', a'' - 1]) +$ votes$(\tau[a + 1, n], \mu'[a + 1, n], [a'', n])$. Together with votes$(\tau[a+1, n], \mu'[a+1, n], [a'', n]) =$ mv$(\chi'[a+1, n], [a'', n])$, we deduce Condition (2) in the lemma statement.

We now consider the if direction. Let τ' be a matching such that $\tau'[a+1, n]$ belongs to MVM$(\chi'[a + 1, n], [a'', n])$ and $\tau'(a) = b$. Since condition (1) holds, we deduce by Lemma 3.3.14 that τ' belongs to reach(χ). Then, we

deduce by Lemma 3.3.15 that

$$\begin{aligned} & \text{votes}(\tau', \mu, [a', n]) \\ = \; & \text{votes}(\mu', \mu, [a', a'' - 1]) + \text{votes}(\tau'[a+1, n], \mu'[a+1, n], [a'', n]) \\ = \; & \text{votes}(\mu', \mu, [a', a'' - 1]) + \text{mv}(\chi'[a+1, n], [a'', n]) \\ = \; & \text{mv}(\chi, [a', n]), \end{aligned}$$

where the second equation follows by the fact that τ' belongs to $\text{MVM}(\chi'[a+1, n], [a'', n])$, and the third equation follows by condition (2). Thus τ' belongs to $\text{MVM}(\chi, [a', n])$, as required. □

Lemma 3.3.17. *Algorithm 3 returns the most preferred object b^* of agent a such that there exists a matching μ' in MVM$(\chi, [a', n])$ with $\mu'(a) = b^*$.*

Proof. It is straightforward to verify that the returned b^* satisfies conditions (1) and (2) in Lemma 3.3.16. Therefore, Lemma 3.3.16 implies that there exists a matching μ' in $\text{MVM}(\chi, [a', n])$ with $\mu'(a) = b^*$.

Next, we prove that b^* is most preferred object of agent a. Assume for the sake of contradiction that there is an object b and a matching μ' such that a prefers b to b^*, μ' belongs to $\text{MVM}(\chi, [a', n])$, and $\mu'(a) = b$. By Lemma 3.3.16, conditions (1) and (2) in Lemma 3.3.16 hold and thus the algorithm does not return b^*, a contradiction. □

We now present Algorithm 4, which uses Algorithm 2 as a building block to find an MVPE matching. In Algorithm 4, we maintain a partial matching τ_i that only involves agents in $[1, i]$. At the ith iteration, we invoke

the BOLA subroutine to find the most preferred object b^* of agent i that satisfies the conditions in Lemma 3.3.17. Next, we apply $\kappa(b^*, i)$ to assign object b^* to agent i. After that, we remove agent i from the configuration.

Let $P(i)$ denote the predicate "there exists a matching ν in $\mathrm{MVM}(\chi, [1, n])$ such that τ_i is a subset of ν". Algorithm 4 maintains the invariant that $P(i)$ holds for all i in $[n]$. This invariant ensures that the final matching τ_n returned by Algorithm 4 is a matching in $\mathrm{MVM}(\chi, [1, n])$. The serial dictatorship mechanism ensures that τ_n is also a Pareto-efficient matching.

Algorithm 4: MVPEM(χ)

Input: A configuration $\chi = (F, \mu)$
Output: An MVPE matching of χ
$\mu_1, a_1, \chi_1, \tau_0 \leftarrow \mu, 1, \chi, \emptyset$
for $i \to 1$ **to** n **do**
  /* i is the index of an agent */
  $b_i \leftarrow \mathrm{BOLA}(\chi_i, a_i)$
  $a'_i \leftarrow \mu_i^{-1}(b_i)$
  $a_{i+1} \leftarrow \max(a'_i + 1, a_i)$
  $\mu' \leftarrow$ the matching that results from applying $\kappa(a'_i, i)$ to μ_i
  /* Assign b_i to agent i and truncate it from χ */
  $\tau_i, \mu_{i+1}, \chi_{i+1} \leftarrow \tau_{i-1} + (i, b_i), \mu'[i + 1, n], (F, \mu_{i+1})$
end
return τ_n

Theorem 3.3.18. *The matching τ_n returned by Algorithm 4 is an MVPE matching.*

Proof. For any two disjoint matchings μ and μ', we write $\mu + \mu'$ to denote the matching $\mu \cup \mu'$. Whenever this notation is used in the argument below, it is straightforward to check that the associated matchings are disjoint.

Below we use induction on i to prove that the following predicates hold for all i in $[n]$.

$P(i)$: there exists $\nu \in \mathrm{MVM}(\chi, [1,n])$ such that τ_i is a submatching of ν;

$Q(i)$: $\mathrm{votes}(\tau_{i-1} + \mu_i, \mu, [1, a_i - 1]) + \mathrm{mv}(\chi_i, [a_i, n]) = \mathrm{mv}(\chi)$.

Notice that by Lemma 3.3.17, for all i in $[n]$, there exists a matching μ'_i in $\mathrm{MVM}(\chi_i, [a_i, n])$ such that $\mu'_i(i) = \tau(i)$.

For $i = 1$, we have $a_1 = 1$, i.e., we deduce $P(1)$ directly from Lemma 3.3.17, and $Q(1)$ from $\tau_0 + \mu_1 = \mu$ and $\mathrm{mv}(\chi_1, [1,n]) = \mathrm{mv}(\chi)$.

Assume that $P(k-1)$ and $Q(k-1)$ hold. We now prove that $P(k)$ and $Q(k)$ hold.

To prove $Q(k)$, it suffices to prove that

$$\begin{aligned} & \mathrm{votes}(\tau_{k-1} + \mu_k, \mu, [1, a_k - 1]) + \mathrm{mv}(\chi_k, [a_k, n]) \\ = \; & \mathrm{votes}(\tau_{k-2} + \mu_{k-1}, \mu, [1, a_{k-1} - 1]) + \mathrm{mv}(\chi_{k-1}, [a_{k-1}, n]), \end{aligned} \tag{3.1}$$

since the right-hard side of the equation is equal to $\mathrm{mv}(\chi)$ by $Q(k-1)$.

We deduce by Lemma 3.3.16 that

$$\mathrm{votes}(\mu'_k, \mu_{k-1}, [a_{k-1}, a_k - 1]) + \mathrm{mv}(\chi_k, [a_k, n]) = \mathrm{mv}(\chi_{k-1}, [a_{k-1}, n]),$$

where $\mu'_k = \{(k-1, \tau_n(k-1))\} + \mu_k$. Thus, to derive Eq. (3.1), it is enough to prove that

$$\begin{aligned} & \mathrm{votes}(\mu'_k, \mu_{k-1}, [a_{k-1}, a_k - 1]) \\ = \; & \mathrm{votes}(\tau_{k-1} + \mu_k, \mu, [1, a_k - 1]) - \mathrm{votes}(\tau_{k-2} + \mu_{k-1}, \mu, [1, a_{k-1} - 1]). \end{aligned}$$

The above equation follows since $\tau_{k-1}+\mu_k$ is reached from applying $\kappa(a'_{k-1}, k-1)$ to $\tau_{k-2}+\mu_{k-1}$.

To prove $P(k)$, notice that Lemma 3.3.17 implies that there is a matching μ'_k in MVM$(\chi_k, [a_k, n])$ such that $\mu'_k(k) = \tau(k)$. Let $\nu = \tau_{k-1} + \mu'_k$. Clearly τ_k is a submatching of ν, and $\nu[k, n] = \mu'_k$ belongs to reach(χ_k). Since rational$(\kappa(a'_k, k))$ holds, Lemma 3.3.14 implies that $\nu[k-1, n]$ belongs to reach(χ_{k-1}). Notice that rational$(\kappa(a'_j, j))$ holds for $j = k-1, \ldots, 2$. Thus, by recursively applying Lemma 3.3.14, we can establish that $\nu[j-1, n]$ belongs to reach(χ_{j-1}) for $j = k-1, \ldots, 2$. Therefore, for $j = 2$, we deduce that $\nu = \nu[1, n]$ belongs to reach(χ_1) = reach(χ). We deduce by Lemma 3.3.15 that

$$\text{votes}(\nu, \mu, [1, n]) = \text{votes}(\nu, \mu, [1, a_k - 1]) + \text{votes}(\mu'_k, \mu_k, [a_k, n]).$$

Moreover, since μ'_k belongs to MVM$(\chi_k, [a_k, n])$,

$$\text{votes}(\mu'_k, \mu_k, [a_k, n]) = \text{mv}(\chi_k, [a_k, n]).$$

By $Q(k)$, we deduce that votes$(\nu, \mu, [1, a_k-1])+\text{mv}(\chi_k, [a_k, n]) = \text{mv}(\chi)$. Thus votes$(\nu, \mu, [1, n]) = \text{mv}(\chi)$, and hence $P(k)$ holds. This completes our proof by induction that $Q(i)$ and $P(i)$ hold for all i in $[n]$.

Since $P(n)$ holds, we deduce that τ_n belongs to MVM(χ). It remains to prove that τ_n is a Pareto-efficient matching. Assume for the sake of contradiction that there is a matching μ^* in reach(χ) such that μ^* Pareto-dominates τ_n. Since τ_n belongs to MVM(χ), it follows by that μ^* also belongs to MVM(χ).

Next, let i^* denote the minimum agent such that $\mu^*(i^*) \neq \tau_n(i^*)$. Thus τ_{i^*-1} is a submatching of μ^* and agent i^* prefers $\mu^*(i^*)$ to $\tau_n(i^*)$ as μ^* Pareto-dominates τ_n. To reach a contradiction, it suffices to show that $\mu^*[i^*, n]$ belongs to $\text{MVM}(\chi_{i^*}, [a_{i^*}, n])$. Once we prove this, we deduce by Lemma 3.3.17 that $\tau_n(i^*)$ is the most preferred object of agent i^* among all objects matched with i^* in some matching in $\text{MVM}(\chi_{i^*}, [a_{i^*}, n])$, a contradiction since agent i^* prefers $\mu^*(i^*)$ to $\tau_n(i^*)$.

To prove that $\mu^*[i^*, n]$ belongs to $\text{MVM}(\chi_{i^*}, [a_{i^*}, n])$, let $R(j)$ denote the predicate "$\mu^*[j, n]$ belongs to $\text{MVM}(\chi_j, [a_j, n])$". It suffices to prove that $R(j)$ holds for all j in $[i^*]$; We prove this claim by induction on j. For $j = 1$, $R(1)$ holds since μ^* belongs to $\text{MVM}(\chi) = \text{MVM}(\chi, [1, n]) = \text{MVM}(\chi, [a_1, n])$. Let $k \leq i^*$ be a positive integer and assume that $R(j)$ holds for $1 \leq j < k$. We need to prove that $R(k)$ holds, i.e., that $\mu^*[k+1, n]$ belongs to $\text{MVM}(\chi_{k+1}, [a_{k+1}, n])$. Notice that $\mu^*[k, n]$ is a matching in $\text{MVM}(\chi_k, [a_k, n])$ such that $\mu^*(k) = \tau_n(k)$. We deduce by Lemma 3.3.15 that $\mu^*[k + 1, n]$ belongs to $\text{MVM}(\chi_{k+1}, [a'', n])$ where $a'' = \max(\mu_k^{-1}(\tau_n(k)) + 1, a_k)$. It is straightforward to verify that $b_k = \tau_n(k)$, $a'_k = \mu_k^{-1}(b_k)$, and $a_{k+1} = \max(a'_k + 1, a_k) = a''$. Therefore, $\mu^*[k + 1, n]$ belongs to $\text{MVM}(\chi_{k+1}, [a'', n]) = \text{MVM}(\chi_{k+1}, [a_k, n])$, which concludes the induction step. □

3.4 Star Networks

In this section, we consider the case where the social network is a star. Our results are twofold. We first present a polynomial-time algorithm for

finding an MVPE matching in a star network when preferences are strict. We then show that finding an MVPE matching in a star network with weak preferences is NP-hard.

For the case of strict preferences, we present a simple polynomial-time algorithm for finding an MVPE matching. The main idea of the algorithm is to use the fact that any swap involves the center agent and a non-center agent to reduce the MVPE problem to the problem for finding a longest path in a directed acyclic graph.

Theorem 3.4.1. *There is a polynomial-time algorithm finding an MVPE matching in a star network with strict preferences.*

Proof. Let O denote the center agent and let $[n]$ denote the set of leaf agents. Let μ denote the initial matching. Note that every swap in a star network involves the center agent. Moreover, notice that any leaf agent can change its object at most once in any matching history. To see this, observe that once an object moves from the center to a leaf, it can never return to the center. Thus, any matching history can be represented as an ordered list of leaf agents. Let $L = (i_1, \ldots, i_k)$ denote such an list, where k belongs to $[n]$. Note that $\mu(i_j) \prec_O \mu(i_{j+1})$ for all j in $[k-1]$, where O is the center agent.

We first show how to find a maximum votes matching, and then prove that any maximum votes matching is also Pareto-efficient.

Finding a maximum votes matching in a star network, it is equivalent to find a longest ordered list of leaves corresponding to a matching history, since

a leaf agent improves its match if and only if it is included in the list. This problem can be reduced to searching for a longest path in a directed acyclic graph $G = (\{O\} \cup [n], E')$, as follows. The edge set E' contains a directed edge from a to a' if agent a' belongs to $[n]$, $\mu(a') \succ_O \mu(a)$, and $\mu(a) \succ_{a'} \mu(a')$. There is a path from O to a' in the digraph G if and only if agent O can be matched to the initial endowment of agent a'. It is straightforward to see that there is a polynomial-time algorithm for solving this longest path problem in G.

Next we prove that any maximum votes matching is Pareto-efficient. Let ν be a maximum votes matching. Assume for the sake of contradiction that reachable matching ν' Pareto-dominates ν. Consider the ordered lists of agents L and L' corresponding to sequences of swaps for reaching ν and ν', respectively. Since ν' Pareto-dominates ν and any agent in L prefers its match in ν to its initial endowment, we deduce that every agent in L belongs to L'. Since ν is a maximum votes matching and any agent in L' prefers its match in ν' to its initial endowment, we deduce that $|L'| \leq |L|$. Since every agent in L belongs to L' and $|L'| \leq |L|$, it follows that the set of agents in L' is equal to the set of agents in L. Since the agents in L and L' are ordered by the preference of the center agent O, we conclude that $L = L'$. Hence $\nu = \nu'$, a contradiction. □

3.4.1 Weak Preferences

In this section, we show that finding an MVPE matching is NP-hard in a star network with weak preferences. We first introduce the notion of a center object sequence. For all matching histories $\Pi = \mu_0, \ldots, \mu_k$, we define the corresponding center object sequence π as the list of objects owned by the center agent O, i.e., π is the sequence $\mu_0(O), \ldots, \mu_k(O)$. We use $\pi[i]$ to denote $\mu_i(O)$, and for any object b, we use $\pi(b)$ to denote the index of the first occurrence of b in π; if b does not belong to π, then $\pi(b)$ is defined to be -1 . We have the following observations for the center object sequence.

Observation 3.4.2. *Let X be a non-center agent with initial endowment x such that x belongs to π. Then $\pi[\pi(x) - 1] \succcurlyeq_X x$.*

Notice that $\pi[\pi(x) - 1]$ is the object used by center agent O to swap x from agent X.

Observation 3.4.3. *Let X be an agent with initial object x. Then agent X is improved by Π if and only if x belongs to π.*

We now prove that it is NP-hard to find a maximum votes matching.

Construction: We use a reduction from 3-SAT. In an instance of 3-SAT, we are given a propositional formula ϕ that is the conjunction of m clauses $C_1, \ldots, C_m$. Each clause C_i is the disjunction of three literals, where each literal is either a variable or the negation of a variable. The set of variables is $x_1, \ldots, x_n$. Given a formula ϕ, we construct a corresponding star configuration

$\chi_\phi = (F, \mu)$ with $3n+3m+1$ agents such that for any 3-SAT formula ϕ with n variables and m clauses, ϕ is satisfiable if and only if χ_ϕ has maximum voting number $2n + 3m$.

We begin by creating the set of agents, which we call A. For each variable index i, there are three agents in A: S_i, T_i, and F_i. For each clause $C_j = \ell_{j,0} \cup \ell_{j,1} \cup \ell_{j,2}$, there are three agents $P_{j,0}$, $P_{j,1}$, and $P_{j,2}$ in A. There is also a center agent O. Thus there are a total of $3m + 3n + 1$ agents in A.

For each agent in A, we use the corresponding lowercase letter to denote its initial object. For example, agent $P_{1,2}$ initially holds object $p_{1,2}$. For agents T_i and F_i in a variable gadget with i in $[n]$, their initial objects are t_i and f_i. Furthermore, for each literal $\ell_{j,k}$, we define $a_{j,k}$ as follows: If $\ell_{j,k}$ is of the form x_w, then $a_{j,k}$ is object t_w; otherwise $a_{j,k}$ is f_w. For example, if $\ell_{j,k} = x_{15}$, then $a_{j,k} = t_{15}$, and if $\ell_{j,k} = \neg x_7$, then $a_{j,k} = f_7$.

For agent preferences, we only consider the objects that each agent likes at least as well as its initial object; the ordering of the remaining objects is immaterial. The center agent O is indifferent to all objects. In specifying the preference list of an agent, we list the objects from most preferred to least preferred, we use a boxed object to denote the initial endowment of the agent, and we use parentheses to group the objects that the agent is indifferent between. The preference list of each agent S_i is $(t_i, f_i), (o, \boxed{s_i})$. The preference list of each agent T_i is $s_i, \boxed{t_i}$. The preference list of each agent F_i is $s_i, \boxed{f_i}$. For each k in $\{0, 1, 2\}$, the preference list of each agent $P_{j,k}$ is $p_{j,k-1}, a_{j,k}, \boxed{p_{j,k}}$, where $p_{j,-1} = p_{j,2}$.

We now present some properties of our gadgets.

Lemma 3.4.4. *Let* Π *denote the matching history* $\mu_0, \ldots, \mu_k$*, let* i *belongs to* $[n]$ *and let* π *denote the corresponding center object sequence of* Π*. Then at most two agents in* $\{S_i, T_i, F_i\}$ *are improved by* Π*. Furthermore, if two agents in* $\{S_i, T_i, F_i\}$ *are improved, then exactly one of* t_i *and* f_i *belongs to* π*.*

Proof. Notice that if agent T_i or F_i is improved, then either T_i or F_i is assigned object s_i, due to the preferences of T_i and F_i. However, in any matching, s_i is matched to a single agent. Therefore Π cannot improve both T_i and F_i. That is, at most two agents in $\{S_i, T_i, F_i\}$ are improved by Π. If exactly two agents in $\{S_i, T_i, F_i\}$ are improved, then exactly one agent in $\{T_i, F_i\}$ is improved, and hence Observation 3.4.3 implies that exactly one of t_i and f_i belongs to π. □

Lemma 3.4.5. *Let* C_j *denote the clause* $\ell_{j,0} \cup \ell_{j,1} \cup \ell_{j,2}$*. For each* k *in* $\{0, 1, 2\}$*, let* $P_{j,k}$ *denote the agent corresponding to literal* $\ell_{j,k}$*, let* $p_{j,k}$ *denote the initial object of* $P_{j,k}$*, and let* $a_{j,k}$ *in* $\{t_i, f_i \mid i \in [n]\}$ *denote the object associated with literal* $\ell_{j,k}$*. Let* Π *denote the matching history* $\mu_0, \ldots, \mu_k$*, let* π *denote the corresponding center object sequence of* Π*, and assume that no object in* $\{a_{j,0}, a_{j,1}, a_{j,2}\}$ *belongs to* π*. Then no object in* $\{p_{j,0}, p_{j,1}, p_{j,2}\}$ *belongs to* π*.*

Proof. We prove the contrapositive. Assume that there exists an object in $\{p_{j,0}, p_{j,1}, p_{j,2}\}$ that belongs to π. Let $p_{j,k}$ denote the object with minimum positive $\pi(p_{j,k})$ where k belongs to $\{0, 1, 2\}$. By Observation 3.4.2, $\pi[\pi(p_{j,k}) -$

1] $\succcurlyeq_{P_{j,k}} p_{j,k}$. By the preferences of $P_{j,k}$, $\pi[\pi(p_{j,k}) - 1]$ belongs to $\{a_{j,k}, p_{j,k-1}\}$ with $p_{j,-1} = p_{j,2}$. Moreover, by the assumption that $p_{j,k}$ is the object with minimum positive $\pi(p_{j,k})$, $\pi[\pi(p_{j,k}) - 1]$ does not belong to $\{p_{j,0}, p_{j,1}, p_{j,2}\}$. Therefore, $\pi[\pi(p_{j,k}) - 1]$ is $a_{j,k}$, and hence π contains object $a_{j,k}$. □

Lemmas 3.4.6 and 3.4.7 below establish the correctness of our reduction.

Using the above properties of our gadgets, it is not hard to verify the correctness of our reduction. Formally, we prove following two lemmas, one for each reduction direction.

Lemma 3.4.6. *If* $\mathrm{mv}(\chi_\phi) = 2n + 3m$*, then* ϕ *is satisfiable.*

Proof. By Lemma 3.4.4, at least n agents are not improved. Moreover, the center agent O initially holds one of its most preferred objects, and hence cannot be improved. Therefore, at most $2n + 3m$ agents are improved. If χ_ϕ achieves the maximum voting number $2n + 3m$, then exactly two agents are improved in each variable gadget and all three agents are improved in each clause gadget.

By Lemma 3.4.4, for all i in $[n]$, exactly one of t_i and f_i belongs to π. We construct an assignment A_π from π as follows: We set x_i to true if t_i belongs to π, and to false otherwise. We claim that A_π is a satisfying assignment for ϕ. Assume for the sake of contradiction that some clause C_j is not satisfied by A_π. Hence, the literals $\ell_{j,0}$, $\ell_{j,1}$, and $\ell_{j,2}$ are evaluated to false under assignment A_π. Recall that $a_{j,0}$, $a_{j,1}$, and $a_{j,2}$ denote the objects associated with literals

$\ell_{j,0}$, $\ell_{j,1}$, and $\ell_{j,2}$, respectively. Thus no object in $\{a_{j,0}, a_{j,1}, a_{j,2}\}$ belongs to π. By Lemma 3.4.5, no object in $\{p_{j,0}, p_{j,1}, p_{j,2}\}$ belongs to π. Therefore, Observation 3.4.3 implies that no agent in $\{P_{j,0}, P_{j,1}, P_{j,2}\}$ is improved by π. □

Lemma 3.4.7. *If there is a satisfying assignment for ϕ, then* $\mathrm{mv}(\chi_\phi) = 2n + 3m$.

Proof. Let A_ϕ be a satisfying assignment for ϕ. For each i in $[n]$, if A_ϕ sets x_i to true, then let b_i denote object t_i; otherwise, let b_i denote object f_i.

Now we describe an iterative procedure for constructing a sequence of swaps to improve $2n + 3m$ agents. The procedure maintains a set $\mathcal{C}$, which is initialized to be the set of all clauses $\{C_1, \ldots, C_m\}$. For each i in $[n]$, let B_i denote the initial owner of b_i, i.e., $B_i = T_i$ if $b_i = t_i$ and $B_i = F_i$ otherwise. We construct the sequence of swaps as follows. For each object b_i, we do the following swaps.

1. We perform two swaps (O, S_i) and (O, B_i). After these two swaps, the center agent O is assigned to b_i and S_i is assigned to object o.

2. If there is a clause C_j in $\mathcal{C}$ and a literal $\ell_{j,k}$ in C_j such that $b_i = a_{j,k}$, then we perform the following steps.

   (a) We perform the sequence of swaps:

$$(O, P_{j,k}), (O, P_{j,k\oplus 1}), (O, P_{j,k\oplus 2}), (O, P_{j,k}),$$

where $\oplus$ denotes addition modulo 3. Remark: After the sequence of swaps, agent $P_{j,k}$ is matched to its most preferred object for all k in $\{0, 1, 2\}$, and center agent O continues to hold b_i.

(b) We remove the clause C_j from $\mathcal{C}$.

3. Finally, we perform the swap (O, S_i). Remark: After this swap, agent O has object o and S_i has one of its most preferred objects b_i. We then proceed to consider object b_{i+1}.

We now show that the sequence of swaps constructed above improves $2n + 3m$ agents. After applying the above sequence of swaps, for all i in $[n]$, agent S_i is matched to object b_i and agent B_i is matched to object s_i. Thus the above sequence of swaps improves $2n$ agents in variable gadgets. We now consider the agents in clause gadgets. Since A_ϕ is a satisfying assignment for ϕ, for any clause C_j, there is a k in $\{0, 1, 2\}$ such that literal $\ell_{j,k}$ is evaluated to true under assignment A_ϕ. Therefore, by the definition of object $a_{j,k}$, we deduce that $a_{j,k}$ is b_i, i.e., agent $P_{j,k}$ is improved. Thus all agents associated with clause C_j are improved. That is, the above sequence of swaps improves all agents in clause gadgets. Since there are m clauses, we conclude that a total of $2n + 3m$ agents are improved. □

Theorem 3.4.8. *Finding an MVPE matching on stars with weak preferences is NP-hard.*

3.5 Generalized Star Networks

In this section, we establish the NP-hardness of finding a maximum votes matching. In particular, we prove Theorem 3.5.1 below by a reduction to reachable object problem on generalized stars, which was shown to be NP-complete by Gourvès et al. [91].

Theorem 3.5.1. *Finding an MVPE matching on generalized stars with strict preferences is NP-hard.*

Proof. We use a reduction from RO on generalized stars. Let $I = (\chi, O, b)$ be an RO instance on generalized stars where $\chi = (F, \mu)$ denotes the configuration on a tree network, and O denotes an agent, and b denotes the target object. Let $I = (\chi, O, b)$ denote an RO instance on generalized stars. Gourvès et. al [91] shows that it is NP-complete to decide whether there is a matching μ' in reach(χ) such that $\mu'(O) = b$.

Let $\chi = (F, \mu)$ where $F = (A, B, \succ, E)$. Let $n = |A|$. We construct a configuration $\chi' = (F', \mu')$ below with $F' = (A', B', \succ', E')$.

We first construct the OAF $F' = (A', B', \succ', E')$. We construct A' by adding $n + 1$ dummy agents $D_1, \ldots, D_{n+1}$, to A, i.e.,

$$A' = A \cup \{D_1, \ldots, D_{n+1}\}.$$

Furthermore, let d_i denote the initial endowment of dummy agent D_i for all i in $[n + 1]$. We construct $B' = B \cup \{d_1, \ldots, d_{n+1}\}$. We construct E' by adding $n + 1$ edges connecting each dummy agent D_i with the agent O into E, i.e.,

$E' = E \cup \{(D_i, O) \mid i \in [n+1]\}$. Clearly, the graph $G' = (A', E')$ is still a tree. In the end, we construct the preference profile $\succ'$ from the preference profile $\succ$. Note that we consider the preferences of each agent as an ordered list starting from its most favorite object to its least favorite object. For convenience, let $d_0 = b$. In the following, we describe the preference lists in $\succ'$ for each agent in A'. For each dummy agent D_i with $i \in [n+1]$, agent D_i prefers only d_{i-1} to its initial endowment d_i, i.e., its preference list consists of d_{i-1}, followed by d_i, and followed by the remaining objects in an arbitrary order. Agent O prefers all dummy objects to all non-dummy objects, and its preference list consists of $d_{n+1}, \ldots, d_1$ in order, followed by its preference list in $\succ$. All other agents prefer all non-dummy objects to all dummy objects, where the ordering of the all non-dummy objects is the same as in B, and the ordering of the dummy objects is arbitrary. Let μ' denote the matching $\mu \cup \{(D_i, d_i) \mid i \in [n+1]\}$.

It is clear that our reduction can be carried out in polynomial time. Hence, it remains only to establish the correctness of our reduction. We begin by proving two useful claims.

Claim 1: Let i belong to $[n+1]$ and let Π denote a matching history for χ' such that there is a matching ν in Π with $\nu(O) = d_i$. Then there is a matching ν' in Π such that $\nu'(O) = b$.

Proof: Assume that Π is of the form $\mu_0, \ldots, \mu_t$. For any k in $[i]$, let $W(k)$ denote the predicate "there is a matching ν in Π such that $\nu(O) = d_{i-k+1}$"; thus $W(1)$ holds. We claim that $W(k)$ holds for $k = 1, \ldots, i$. For the base case, it is immediate that $W(1)$ holds. For the induction step, fix k in

$\{2, \ldots, i\}$ and assume that $W(k)$ holds. We need to prove that $W(k+1)$ holds. Let j in $[t]$ be the minimum index in $[t]$ such that $\mu_j(O) = d_{i-k+1}$. Since agent O is the only agent adjacent to dummy agent D_{i-k+1} and the initial endowment of D_{i-k+1} is d_{i-k+1}, we deduce that matching μ_j is reached from μ_{j-1} by performing a swap between agents O and D_{i-k+1}. Hence $\mu_{j-1}(D_{i-k+1}) = d_{i-k+1}$, $\mu_j(D_{i-k+1}) = \mu_{j-1}(O)$, and agent D_{i-k+1} prefers $\mu_{j-1}(O)$ to d_{i-k+1}. Since D_{i-k+1} only prefers d_{i-k} to its initial endowment d_{i-k+1}, we deduce that $\mu_{j-1}(O) = D_{i-k}$, i.e., that $W(k+1)$ holds. This completes the proof of induction step. Therefore, for $k = i$, $W(i)$ implies that there exists a matching ν in Π such that $\nu(O) = d_0 = b$. This completes the proof of Claim 1.

Claim 2: Let i belongs to $[n+1]$. Let Π denote a matching history for χ' such that there are two matchings μ^* and μ^{**} in Π with $\mu^*(D_i) \neq \mu^{**}(D_i)$. Then there is a matching ν in Π such that $\nu(O) = b$.

Proof: Let j in $[t]$ be the minimum index such that μ_{j-1} and μ_j are two matchings in Π with $\mu_{j-1}(D_i) \neq \mu_j(D_i)$. Therefore, agent D_i prefers $\mu_j(D_i)$ to $\mu_{j-1}(D_i)$. Since D_i only prefers d_{i-1} to its initial endowment d_i, we deduce that $\mu_i(D_i) = d_{i-1}$. Since agent O is the only agent adjacent to D_i, we deduce that there is a matching μ_k in Π with $k < j$ such that $\mu_k(O) = d_{i-1}$. Moreover, we deduce by Claim 1 that there is a matching ν' in Π such that $\nu'(O) = b$. This completes the proof of Claim 2.

The following claim establishes the correctness of our reduction.

Claim 3: Let Π denote a matching history for χ' that improves $\mathrm{mv}(\chi')$

agents. There is a matching ν in Π such that $\nu(O) = b$ if and only if $\mathrm{mv}(\chi') \geq n + 1$.

Proof: We begin by considering the if direction. Assume that $\mathrm{mv}(\chi') \geq n + 1$. We deduce that some dummy agent D_i is improved by Π. That is, there are two matchings μ^* and μ^{**} in Π such that $\mu^*(D_i) \neq \mu^{**}(D_i)$. Thus, we deduce by Claim 2 that there is a matching ν in Π such that $\nu(O) = b$.

We now consider the only if direction. We first prove that $\nu(D_i) = d_i$ for all i in $[n+1]$. Assume for the sake of contradiction that there is an index i in $[n+1]$ such that $\nu(D_i) \neq d_i$. Note that agent O is the only agent adjacent to of D_i and D_i only prefers d_{i-1} to d_i. It follows that there is a matching ν' in Π such that O prefers $\nu'(O)$ to $\nu(O)$ and $\nu'(O) = d_i$, a contradiction since agent O prefers d_i to $b = \nu(O)$. Therefore, we conclude that $\nu(O) = b$ and $\nu(D_i) = d_i$ for all i in $[n+1]$. Thus, starting from ν, it is straightforward to verify that the following sequence of exchanges is a matching history: $(O, D_1), \ldots, (O, D_{n+1})$. Moreover, this yields a matching history for χ' that improves all $n+1$ dummy agents. Since Π improves the maximum voting number of agents and there is a matching history that improves all $n+1$ dummy agents, we deduce that the number of agents improved by Π is at least $n + 1$. This completes the proof of Claim 3. $\square$

Chapter 4

Fractional Hedonic Games With a Limited Number of Coalitions

4.1 Introduction

Community detection in social networks, or network partitioning, is an important topic in social network analysis. A social network is classically represented by a directed weighted graph over the agents, where a weighted link models the relationship between two agents in the social network. Intuitively, communities correspond to groups of vertices that are internally more densely connected than with the rest of the network. Partitioning a social network into disjoint communities, or revealing the hidden community structure, can offer insights regarding the organization of a social network and can significantly simplify the network representation. Furthermore, in online marketing, such as placing online ads or deploying viral marketing strategies, identifying communities often leads to more accurate targeting and better marketing results.

A key challenge of community detection is to formally define the notion of a community. Various attempts from various perspectives have emerged

The results presented in this chapter are based on my single-authored paper [116].

in the literature (see [80, 106] for two recent surveys on this topic). Many attempts focus on optimizing one of a number of global metrics designed to quantitatively measure the quality of a community structure. In this chapter, we focus on metrics that tend to capture the natural forces and dynamics underlying the formation of communities.

The field of game theory focuses on interactions between intelligent agents. Thus, it is natural to apply game theory to capture the dynamics behind the formation of communities in social networks. In recent work, there has been a considerable amount of research on using game-theoretic techniques to study community detection in social networks. We refer to [99] for a recent survey on this topic. Hedonic games are a notable type of game for studying coalition formation (see [22] for a survey). A hedonic game is specified by a set of players who have preferences over the set of all possible partitions of the players into coalitions. The outcome of a hedonic game consists a partition of the players into disjoint coalitions. Of particular relevance to the present chapter is the line of research initiated by Aziz et al. [20] on using fractional hedonic games to study community detection. Fractional hedonic games (FHGs), introduced by Aziz et al. [18], are a subclass of hedonic games that can be represented by directed weighted graphs. In particular, an FHG is represented by a directed weighted graph where the weight of edge (i, j) denotes the value player i has for player j and the utility of a player i is defined as the average value that player i ascribes to the members of i's coalition. Outcomes that satisfy some notion of stability or welfare are considered to be

desirable community structures for a given FHG. For example, consider FHGs represented by undirected unweighted graphs, i.e., undirected graphs where each edge has weight 0 or 1. This covers situations in which players only distinguish between friends and non-friends, and each play wants to belong to a coalition in which the fraction of friends is maximized. Aziz et al. [20] consider the computational complexity of computing welfare maximizing partitions for FHGs. Three different notions of social welfare are considered: (1) utilitarian welfare (or social welfare), which is based on the sum of utilities; (2) egalitarian welfare, which is based on the minimum utility of any agent; (3) Nash welfare, which is based on the product of utilities. They show that maximizing utilitarian, egalitarian, or Nash welfare is NP-hard even for the FHGs represented by undirected unweighted graphs. On the positive side, they present approximation algorithms. These approximation algorithms, which are based on computing maximal matchings, produce coalitions of size at most two. This limits the practical applicability of these algorithms, as it is often undesirable to form many tiny coalitions. In this chapter, we focus on utilitarian welfare.

Our results. We study a variant of FHGs where there is a specific upper bound k on the number of coalitions that can be formed. To motivate this work, note that in many real-world scenarios, there are physical and organizational restrictions that limit the number of possible coalitions. For example, consider a setting in which each coalition requires a leader, and only a small number of agents are qualified to act as a leader. In such a setting, no feasible partition can contain more coalitions than the number of qualified

leaders.

A central concern of coalition formation games is to define what constitutes a desired outcome. Within our setting, we consider two key objectives. Our first objective is to find a partition maximizing the social welfare, i.e., the sum of the utilities of all players. As mentioned before, computing social welfare maximizing partitions (with no restriction number of coalitions) has been shown to be NP-hard by Aziz et al. [20], even for FHGs represented by undirected unweighted graphs. Here we study the parameterized complexity of the problem in terms of k. We refer to a partition with exactly k coalitions as a k-partition. For all $k \geq 2$, we establish the NP-hardness of finding a social welfare maximizing k-partition on undirected unweighted graphs. For undirected unweighted trees, we prove a structural property of social welfare maximizing k-partitions. In particular, for undirected unweighted trees, we show that every coalition in a social welfare maximizing k-partition is connected. By leveraging this property, we present a simple algorithm for finding a social welfare maximizing k-partition in polynomial time when the underlying social network is an n-node undirected unweighted trees and parameter k is fixed.

A social welfare maximizing partition may admit a player whose utility can be increased by deviating to another coalition. Our second objective is to consider Nash stable partitions, where no player can improve their utility by unilaterally changing their coalition. We prove that for all $k \geq 2$, if a Nash stable k-partition exists in an FHG represented by an n-node directed

unweighted graph with bounded maximum out-degree, then each coalition in such a k-partition is of size $\Omega(n)$. We then study the computational complexity of finding a Nash stable k-partition. Unfortunately, for all $k \geq 2$, we prove that it is NP-complete to determine whether an FHG played on a directed weighted graph with edge weights in $\{0, -1\}$ admits a Nash stable k-partition.

Related works. Aziz et al. [20] studied FHGs from a social welfare perspective, Subsequently, Flammini et al. [77] investigated how to form welfare maximizing coalitions in FHGs in an online setting. Chen et al. [51] proposed several simulation-based methods for finding social welfare maximizing partitions, and provided numerical results. Bilò et al. [34] initiated the study of Nash stable partitions in FHGs from a non-cooperative point of view. They showed that a Nash stable partition is not guaranteed to exist in FHGs played on undirected graphs with negative weights. However, they proved that such a partition always exists when weights are non-negative. Furthermore, they give bounds on the (Nash) price of anarchy and stability. In addition, they established the NP-hardness of computing a Nash stable partition with maximum social welfare. Further results on the price of stability for FHGs played on undirected unweighted graphs have been presented by Kaklamanis et al. [100]. Other stability concepts in FHGs have also been studied [17, 43, 48].

The restriction on the number of coalitions, which is the focus of the present chapter, has been mostly overlooked. Skibski et al. [157] studied k-coalitional cooperative games in the transferable utility setting, and developed an extension of the Shapley value for this game. Additively separable hedonic

games (ASHGs) [18] and modified fractional hedonic games (MFHGs) [137] are two related classes of hedonic games that can be represented by graphs. Sless et al. [158] initiated the study of ASHGs in which exactly k coalitions must be formed. Estivill-Castro et al. [74] studied MFHGs where k equal-size coalitions must be formed (a balanced k-partition).

Aziz et al. [17] explained why efficient or stable outcomes of FHGs tend to provide better partitions than their counterparts for ASHGs and MFHGs, respectively. Sless et al. [158] considered social welfare maximizing partitions and k-coalitions-core stable partitions, the latter being an adaptation of the notion of core stability to their setting. They presented an efficient algorithm for finding social welfare maximizing k-partitions in ASHGs played on undirected graphs when the number of negative-weight edges is limited, and proved that for all $k \geq 1$, it is NP-hard to determine (1) whether a given k-partition is k-coalitions-core and (2) whether there exists a k-coalitions-core stable k-partition in ASHGs. Estivill-Castro et al. [74] considered Nash stability. They proved that for all $k \geq 2$, finding a balanced Nash stable k-partition is NP-hard in general undirected unweighted graphs, but polynomially solvable in undirected unweighted trees. Further results on Nash stable 2-partitions for MFHGs have been presented in [23, 24].

Chapter organization. The remainder of this chapter is organized as follows. Section 4.2 presents formal definitions. Sections 4.3 and 4.4 present our results for social welfare maximizing k-partitions and Nash stable k-partitions, respectively.

4.2 Preliminaries

For all positive integers n, let $[n]$ denote $\{1, \ldots, n\}$. In an FHG, we are given a set $N = [n]$ of players. The objective of the game is to partition the players into disjoint coalitions $\mathcal{P} = \{P_1, P_2, \ldots\}$. Let $\Pi(N)$ denote the set of all partitions of N, and for all integers $k \geq 1$, let $\Pi_k(N)$ denote the set of partitions in $\Pi(N)$ with exactly k coalitions. We refer to each partition in $\Pi_k(N)$ as a k-partition.

Each player i has a value function $v_i : N \to \mathbb{R}$ that denotes how much player i values each of the players in N. We assume that $v_i(i) = 0$. Hence, every FHG can be represented by a tuple of valuation functions $v = (v_1, \ldots, v_n)$. We often associate an FHG with a weighted directed graph. Given a tuple of valuation functions v, let $G = (N, E, v)$ denote the weighted directed graph where the weight of the pair (i, j) in $N \times N$ is $v_i(j)$ and E contains all directed edges corresponding to pairs with non-zero weights. Let $\mathcal{G}(G)$ denote the fractional hedonic game associated with G. We say that a graph G is unweighted if each edge in G has weight 1, and we represent such an unweighted graph as a pair (N, E).

For convenience, for each player i, we extend the input domain of the value function v_i to $N \cup \{P \mid P \subseteq N \wedge i \in P\} \cup \Pi(N)$. For any coalition $P \subseteq N$ that contains player i, the utility $v_i(P)$ of agent i is defined as $\frac{\sum_{j \in P} v_i(j)}{|P|}$. For any partition $\mathcal{P}$ in $\Pi(N)$, let $\mathcal{P}(u)$ denote the coalition that contains player u and let the shorthand $v_i(\mathcal{P})$ denote $v_i(\mathcal{P}(i))$.

Given an FHG $\mathcal{G}(G)$ and a partition $\mathcal{P}$, the social welfare $\mathrm{SW}_{\mathcal{G}(G)}(\mathcal{P})$ of $\mathcal{P}$ is defined as the sum of the utilities of all players, i.e., $\mathrm{SW}_{\mathcal{G}(G)}(\mathcal{P}) = \sum_{i \in N} v_i(\mathcal{P})$. We often drop the subscript $\mathcal{G}(G)$ when there is no ambiguity.

Given a partition $\mathcal{P}$ in $\Pi(N)$, we say that a player i is Nash stable for $\mathcal{P}$ if there is no other coalition $P' \neq \mathcal{P}(i)$ in $\mathcal{P}$ such that $v_i(P' \cup \{i\}) > v_i(\mathcal{P})$. We say that a partition $\mathcal{P}$ in $\Pi(N)$ is Nash stable if all players are Nash stable for $\mathcal{P}$.

We use the following notations from graph theory. Let $G = (N, E)$ be an unweighted graph. Given a subset U of N, we denote by $E_G(U)$ the set of edges of G having both endpoints in U. Moreover, for two disjoint sets N_1 and N_2, we denote by $E_G(N_1, N_2)$ the set of edges having one endpoint in N_1 and one endpoint in N_2. We drop the subscript G when there is no ambiguity. For any graph $G = (N, E)$ and any subset S of N, we let $G[S]$ denote the subgraph of graph G induced by S.

We now state the two problems studied in this chapter.

- The social welfare maximizing k-partition problem: Given a fractional hedonic game $\mathcal{G}(G)$ and an integer k, find a k-partition $\mathcal{P}$ in $\Pi_k(N)$ that maximizes the social welfare $\mathrm{SW}(\mathcal{P})$.

- The Nash stable k-partition problem: Given a fractional hedonic game $\mathcal{G}(G)$ and an integer k, determine whether there is a k-partition $\mathcal{P}$ in $\Pi_k(N)$ that is Nash stable.

4.3 Social Welfare Maximizing k-Partitions

In this section, we focus on FHGs played on undirected unweighted graphs. We remark that for an FHG played on an undirected unweighted graph $G = (N, E)$, the social welfare of any partition $\mathcal{P}$ in $\Pi(N)$ is $\mathrm{SW}_{\mathcal{G}(G)}(\mathcal{P}) = \sum_{P \in \mathcal{P}} \frac{2|E_G(P)|}{|P|}$. For FHGs played on undirected unweighted graphs, Section 4.3.1 establishes the NP-hardness of the social welfare maximizing k-partition problem for every fixed $k \geq 2$. For FHGs played on undirected unweighted trees, Section 4.3.2 presents an efficient algorithm that solves the social welfare maximizing k-partition problem for all $k \geq 2$.

4.3.1 NP-Hardness Results

In this section, we prove Theorem 4.3.1 below.

Theorem 4.3.1. *For FHGs played on unweighted undirected graphs, the social welfare maximizing k-partition problem is NP-hard for every fixed $k \geq 2$.*

We separate our hardness proof into two parts: for $k \geq 3$ and for $k = 2$.

When $k \geq 3$, we reduce from the k-colorable problem, which was proved to be NP-complete by Leven and Galil [115] for all $k \geq 3$. The k-colorable problem asks whether a given undirected graph can be partitioned into k independent sets. By considering complementary graphs, the NP-completeness of the k-colorable problem implies the NP-completeness of determining whether an undirected graph can be partitioned into k cliques. It is straightforward to verify that the social welfare of a k-partition $\mathcal{P}$ for an undirected unweighted

graph is at most $2(n-k)$. Moreover, if a k-partition $\mathcal{P}$ has social welfare $2(n-k)$, then the k-partition $\mathcal{P}$ partitions G into k cliques. Therefore, if there is an efficient algorithm that solves the social welfare maximizing k-partition problem, then there is an efficient algorithm that solves the NP-complete problem of determining whether an undirected graph can be partitioned into k cliques. Thus the social welfare maximizing k-partition problem is NP-hard for $k \geq 3$.

It remains to address the case $k = 2$. We reduce from the max cut problem, which was proved to be NP-complete by Karp [102]. Recall that in the max cut problem, we are given an instance of an unweighted undirected graph G and a positive integer r, and we wish to determine whether there exists a cut (S_1, S_2) of G such that $|E(S_1, S_2)| \geq r$. We remark that our reduction is similar to a reduction given by Bonsma et al. [39]. Bonsma et al. use a reduction from the max cut problem to prove that it is NP-hard to find a cut (S_1, S_2) in an undirected graph G such that $\frac{|E_G(S_1,S_2)|}{|S_1||S_2|}$ is minimized.

Reduction. Let $I = (G, r)$ denote an instance of the max cut problem, where $G = (V, E)$ denotes an undirected graph and r denotes a positive integer. Below we construct an undirected unweighted graph $G^* = (V^*, E^*)$ that represents a corresponding instance of the social welfare maximizing 2-partition problem. For convenience, let n denote $|V|$ and let m denote $|E|$.

We first construct an undirected graph $G' = (V', E')$, and then we let $G^* = (V', K' \setminus E')$ be the complementary graph of G', where K' consists of all 2-element subsets of V'. For each v in V, we have two sets I_v and I'_v of vertices,

each of size $M = 4m + 2$. Thus, G' has $2nM$ vertices and $V' = \bigcup_{v \in V} I_v \cup I'_v$. For each v in V, we introduce edges connecting each vertex in I_v to each vertex in I'_v. We select one distinguished vertex from each I_v (resp., I'_v) to form a set A (resp., A') of n vertices. For each edge (u, v) in E, we add an edge to E' between the two distinguished vertices in I_u and I_v (resp., I'_u and I'_v). The resulting graph is G'.

To show that our reduction is correct, we first prove two useful properties of social welfare maximizing k-partitions. After that, we prove Lemma 4.3.4, which establishes the correctness of our reduction.

Lemma 4.3.2. *Let $H = (V, E)$ be an undirected graph, let $\overline{H}$ be the complementary graph of H, and let $\mathcal{P}$ be a 2-partition in $\Pi(V)$. Then $\mathrm{SW}_{\mathcal{G}(H)}(\mathcal{P}) = |V| - 2 - \mathrm{SW}_{\mathcal{G}(\overline{H})}(\mathcal{P})$.*

Proof. Let n denote $|V|$. Note that $\mathrm{SW}_{\mathcal{G}(H)}(\mathcal{P}) = \sum_{i \in \{1,2\}} \frac{2E_H(P_i)}{|P_i|}$. Since $\overline{H}$ is the complementary graph of H, for all subsets P of V, we have $E_H(P) + E_{\overline{H}}(P) = |P|(|P| - 1)/2$. Therefore,

$$\begin{aligned}
\underset{\mathcal{G}(H)}{\mathrm{SW}}(\mathcal{P}) &= \sum_{i \in \{1,2\}} \left(|P_i| - 1 - \frac{2E_{\overline{H}}(P_i)}{|P_i|} \right) \\
&= n - 2 - \sum_{i \in \{1,2\}} v_{\overline{H}}(P_i) \\
&= n - 2 - \underset{\mathcal{G}(\overline{H})}{\mathrm{SW}}(\mathcal{P}).
\end{aligned}$$

□

Lemma 4.3.3. *Let $\mathcal{P}$ in $\Pi_2(V')$ be a social welfare minimizing 2-partition for G'. Then for each node v in V and each coalition P in $\mathcal{P}$, either $P \cap (I_v \cup I'_v) = I_v$ or $P \cap (I_v \cup I'_v) = I'_v$.*

Proof. Assume for the sake of contradiction that there is a vertex v in V such that the 2-partition $\mathcal{P}$ does not partition $I_v \cup I'_v$ into the sets I_v and I'_v. Let $\mathcal{P} = (P_1, P_2)$. Therefore, for each i in $\{1, 2\}$, both $P_i \cap I_v$ and $P_i \cap I'_v$ are non-empty. Let $a = |P_1 \cap I_v| \geq 1$ and $a' = |P_1 \cap I'_v| \geq 1$. Note that $|E(P_1 \cap I_v, P_1 \cap I'_v)| = aa'$, and $|E(P_2 \cap I_v, P_2 \cap I'_v)| = (M-a)(M-a')$. Without loss of generality, assume that $|E(P_1 \cap I_v, P_1 \cap I'_v)| \geq |E(P_2 \cap I_v, P_2 \cap I'_v)|$, i.e., $aa' \geq (M-a)(M-a')$. That is, $a+a' \geq M$, and hence $aa' \geq a(M-a) \geq M-1$, as a and a' are positive integers. Thus $|E(P_1 \cap I_v, P_1 \cap I'_v)| = aa' \geq M - 1$.

Therefore, we obtain

$$\begin{aligned}
\underset{\mathcal{G}(G')}{\mathrm{SW}}(\mathcal{P}) &= \sum_{i \in \{1,2\}} \frac{2E(P_i)}{|P_i|} \\
&\geq \frac{2E(P_1)}{|P_1|} \\
&\geq \frac{2|E(P_1 \cap I_v, P_1 \cap I'_v)|}{|P_1|} \\
&\geq \frac{M-1}{|P_1|} \\
&\geq \frac{M-1}{2nM}.
\end{aligned}$$

Let $P' = (\bigcup_{v \in V} I_v, \bigcup_{v \in V} I'_v)$. Thus

$$\begin{aligned} \underset{\mathcal{G}(G')}{\mathrm{SW}}(\mathcal{P}) &\leq \underset{\mathcal{G}(G')}{\mathrm{SW}}(P') \\ &= \frac{2|E_{G'}(\bigcup_{v \in V} I_v)|}{nM} + \frac{2|E_{G'}(\bigcup_{v \in V} I'_v)|}{nM} \\ &= \frac{2m}{nM}. \end{aligned}$$

It follows that $\frac{M-1}{2nM} \leq \frac{2m}{nM}$. Rearranging, we obtain $M - 1 \leq 4m$, a contradiction since $M = 4m + 2$. □

Lemma 4.3.4. *At least r edges cross some cut of G if and only if G^* has a 2-partition with social welfare at least $2nM - 2 - \frac{4m-4r}{nM}$.*

Proof. By Lemma 4.3.2, it suffices to prove that at least r edges cross some cut of G if and only if G' has a 2-partition with social welfare at most $\frac{4m-4r}{nM}$. We consider two cases.

Case 1: G has a cut (S_1, S_2) with $|E(S_1, S_2)| \geq r$. For each i in $\{1, 2\}$, let P_i denote $\{I_v \mid v \in S_i\}$ and let P'_i denote $\{I'_v \mid v \in S_i\}$. Consider the 2-partition $\mathcal{P}' = (P_1 \cup P'_2, P'_1 \cup P_2)$. It is straightforward to verify that $|P_1 \cup P'_2| = |P'_1 \cup P_2| = nM$. Moreover, notice that

$$\begin{aligned} |E_{G'}(P_1 \cup P'_2)| &= |E_{G'}(P_1)| + |E_{G'}(P'_2)| \\ &= |E_G(S_1)| + |E_G(S_2)| \\ &= |E| - |E_G(S_1, S_2)| \\ &= m - r. \end{aligned}$$

Similarly, we have $|E_{G'}(P_1' \cup P_2)| = m - r$. Thus

$$\mathrm{SW}(\mathcal{P}') = \frac{2|E_{G'}(P_1 \cup P_2')|}{|P_1 \cup P_2'|} + \frac{2|E_{G'}(P_1' \cup P_2)|}{|P_1' \cup P_2|} = \frac{4m - 4r}{nM}. \tag{4.1}$$

Therefore, G' has a 2-partition with social welfare at most $\frac{4m-4r}{nM}$.

Case 2: G' has a 2-partition with social welfare at most $\frac{4m-4r}{nM}$. Consider a social welfare minimizing 2-partition $\mathcal{P}^* = (P_1^*, P_2^*)$ in G'. Hence $\mathrm{SW}_{\mathcal{G}(G')}(P^*) \leq \frac{4m-4r}{nM}$. By Lemma 4.3.3, we deduce that for each v in V, $\mathcal{P}^*$ partitions $I_v \cup I_v'$ into the two sets I_v and I_v'. Therefore, either $I_v \subseteq P_1^*$ and $I_v' \subseteq P_2^*$, or $I_v \subseteq P_2^*$ and $I_v' \subseteq P_1^*$. For each i in $\{1, 2\}$, let S_i denote $\{v \in V \mid I_v \subseteq P_i^*\}$ and S_i' denote $\{v \in V \mid I_v' \subseteq P_i^*\}$. Clearly, $S_1 = S_2'$, $S_2 = S_1'$, and $(S_1, S_1') = (S_2', S_2)$ is a partition in $\Pi_2(V)$. Using a calculation similar to that used to derive Eq. (4.1), we have $\mathrm{SW}_{\mathcal{G}(G')}(\mathcal{P}^*) = \frac{4(m-|E_G(S_1,S_1')|)}{nM}$. Since $\mathrm{SW}_{\mathcal{G}(G')}(P^*) \leq \frac{4m-4r}{nM}$, we deduce that $|E_G(S_1, S_1')| \geq r$. Therefore, at least r edges cross some cut of G. □

4.3.2 Finding Social Welfare Maximizing k-Partitions for Trees

In this section, we first prove Lemma 4.3.5, which presents a useful structural property of social welfare maximizing k-partitions on undirected unweighted trees. Then, based on this property, for any fixed positive integer k, we present a simple polynomial time algorithm for the social welfare maximizing k-partition problem on unweighted undirected trees.

Lemma 4.3.5. *Let $G = (N, E)$ be a tree, let k belong to $[|N|]$, and let $\mathcal{P}^*$ be a social welfare maximizing k-partition in $\Pi_k(N)$ with coalitions $P_1^*, \ldots, P_k^*$.*

Then for all coalitions P_i^* in $\mathcal{P}^*$, $G[P_i^*]$ is connected.

Proof. Assume for the sake of contradiction that there is a coalition P_i^* in $\mathcal{P}^*$ such that $G[P_i^*]$ is not connected. Thus $G[P_i^*]$ has $p \geq 2$ connected components. Let $T_1, \ldots, T_p$ denote the vertex sets associated with these p connected components. Hence each connected component $G[T_1], \ldots, G[T_p]$ of $G[P_i^*]$ is a tree. Observe that $|P_i^*| = \sum_{j \in [p]} |T_j|$. Without loss of generality, assume that $|T_1| \leq \cdots \leq |T_p|$.

Since G is a tree, we deduce that there is another coalition P_j^* in $\mathcal{P}^*$ with $j \neq i$ such that there is an edge between P_j^* and T_1. Below we construct a k-partition $\mathcal{P}'$ in $\Pi_k(N)$ with $\mathrm{SW}(\mathcal{P}') > \mathrm{SW}(P^*)$, a contradiction since $\mathcal{P}^*$ is a social welfare maximizing k-partition.

We construct such a k-partition $\mathcal{P}'$ with coalitions $P_1', \ldots, P_k'$ as follows. For each ℓ in $[k] \setminus \{i, j\}$, we set P_ℓ' to P_ℓ^*. Furthermore, we set P_i' to $P_i^* \setminus T_1$ and P_j' to $P_j^* \cup T_1$. Now we prove that $\mathrm{SW}(\mathcal{P}') > \mathrm{SW}(\mathcal{P}^*)$. For any coalitions S of N, let $d(S)$ denote $E(S)/|S|$. Thus, $\mathrm{SW}(\mathcal{P}') = \sum_{w \in [k]} 2d(P_w')$ and $\mathrm{SW}(\mathcal{P}^*) = \sum_{w \in [k]} 2d(P_w^*)$. Hence it suffices to prove that $d(P_i') + d(P_j') > d(P_i^*) + d(P_j^*)$. Below we establish the preceding inequality by showing that $d(P_i') \geq d(P_i^*)$ and $d(P_j') > d(P_j^*)$.

First, we prove that $d(P_i') \geq d(P_i^*)$. Recall that the subgraph $G[P_i^*]$ contains p trees $G[T_1], \ldots, G[T_p]$, while the subgraph $G[P_i']$ contains trees $G[T_2], \ldots, G[T_p]$. Thus

$$d(P_i') = \frac{\sum_{\ell=2}^{p} (|T_\ell| - 1)}{\sum_{\ell=2}^{p} |T_\ell|} = 1 - \frac{p-1}{|P_i^*| - t_1}$$

and

$$d(P_i^*) = \frac{\sum_{\ell=1}^{p}(|T_\ell| - 1)}{\sum_{\ell=1}^{p}|t_\ell|} = 1 - \frac{p}{|P_i^*|}.$$

Now, to show that $d(P_i') \geq d(P_i^*)$, it suffices to show that $\frac{p-1}{|P_i^*|-t_1} \leq \frac{p}{|P_i^*|}$, i.e., $p \cdot t_1 \leq |P_i^*|$. The latter inequality follows from $|T_1| \leq \cdots \leq |T_p|$ and $|P_i^*| = \sum_{\ell=1}^{p}|T_\ell|$.

Second, we prove that $d(P_j') > d(P_j^*)$. Since P_j^* is a forest, we have $|E(P_j^*)| < |P_j^*|$. Since there is at least one edge connecting P_j^* and T_1, we have

$$d(P_j') = \frac{|E(P_j^* \cup T_1)|}{|P_j^*| + |T_1|} \geq \frac{|E(P_j^*)| + 1 + |E(T_1)|}{|P_j^*| + |T_1|} = \frac{|E(P_j^*)| + |T_1|}{|P_j^*| + |T_1|}.$$

Since $|E(P_j^*)| < |P_j^*|$, we conclude that $d(P_j') > |E(P_j^*)|/|P_j^*| = d(P_j^*)$. □

Theorem 4.3.6. *For any fixed positive integer k, the social welfare maximizing k-partition problem can be solved in polynomial time on undirected unweighted trees.*

Proof. By Lemma 4.3.5, we deduce that any social welfare maximizing k-partition on a tree is a partition into k subtrees. Notice that for trees, there is a one-to-one correspondence between removing $k-1$ edges and partitioning into k subtrees. Let n denote the number of players in G. Thus, for any fixed k, we can use brute force to evaluate each of the $\binom{n-1}{k-1}$ k-partitions of G in polynomial time. □

4.4 Nash Stable k-Partitions

In this section, we consider Nash stable k-partitions for all $k \geq 2$. Throughout this section, we assume that $k \geq 2$ unless otherwise stated. As an independent result, Theorem 4.4.1 below shows that a Nash stable k-partition of an unweighted directed graph with bounded out-degree is almost balanced. Then, we prove that it is NP-complete to determine whether a directed weighted graph where all edges have weights -1 admits a Nash stable k-partition.

Theorem 4.4.1. *Let $k \geq 2$ and $\Delta \geq 2$ be integers, let $G = (N, E)$ denote a directed unweighted strongly connected graph with out-degree at most Δ and $|N| \geq k \cdot \Delta^{k+1}$, and assume that $\mathcal{G}(G)$ admits a Nash stable k-partition $\mathcal{P}$ in $\Pi_k(N)$. Then all coalitions in $\mathcal{P}$ have size at least $\frac{n}{k \cdot \Delta^{k-1}}$.*

Proof. Let $P_1, \ldots, P_k$ denote the k coalitions in $\mathcal{P}$ with $0 < |P_1| \leq \cdots \leq |P_k|$. It suffices to prove that $|P_1| \geq \frac{n}{k \cdot \Delta^{k-1}}$. For any t in $\{0, \ldots, k-1\}$, let $Q(t)$ denote the predicate $|P_{k-t}| \geq \frac{n}{k \cdot \Delta^t}$. We use induction on t to prove that $Q(t)$ holds for any t in $\{0, \ldots, k-1\}$. Clearly, $Q(k-1)$ implies that $|P_1| \geq \frac{n}{k \cdot \Delta^{k-1}}$.

For the base case, notice that $0 < |P_1| \leq \cdots \leq |P_k|$ and hence $|P_k| \geq \frac{1}{k} \sum_{i \in [k]} |P_i| = \frac{n}{k}$. Therefore, $Q(0)$ holds. For the induction step, let i in $[k-1]$ be given and suppose that $Q(t)$ holds for each t in $\{0, \ldots, i-1\}$. We need to prove that $Q(i)$ holds. Since the partition $P_1, \ldots, P_k$ belongs to $\Pi_k(N)$, we deduce that $(\bigcup_{j \in [k-i]} P_j, \bigcup_{j \in [k] \setminus [k-i]} P_j)$ is a 2-partition in $\Pi_2(N)$. Furthermore, since G is strongly connected, there is a directed edge (b, a) from

$\bigcup_{j\in[k]\setminus[k-i]} P_j$ to $\bigcup_{j\in[k-i]} P_j$. Let integer i' (resp., j') be such that player a (resp., b) belongs to $P_{i'}$ (resp, $P_{j'}$). Thus $1 \leq i' \leq k-i$ and $k-i+1 \leq j' \leq k$. Therefore, $|P_{i'}| \leq |P_{k-i}|$ and $k - j' \leq i - 1$. By the induction hypothesis, we know that $Q(k-j')$ holds, i.e., $|P_{j'}| \geq \frac{n}{k\Delta^{k-j'}} \geq \frac{n}{k\Delta^{i-1}}$. Below we prove that $|P_{i'}| \geq \frac{1}{\Delta}|P_{j'}|$. Since $|P_{i'}| \leq |P_{k-i}|$ and $|P_{j'}| \geq \frac{n}{k\Delta^{i-1}}$, we deduce that $|P_{k-i}| \geq |P_{i'}| \geq \frac{1}{\Delta}|P_{j'}| \geq \frac{1}{\Delta} \cdot \frac{n}{k\Delta^{i-1}} = \frac{n}{k\Delta^{i}}$. Hence $Q(i)$ holds, as required.

It remains to prove that $|P_{i'}| \geq \frac{1}{\Delta}|P_{j'}|$. Since $\mathcal{P}$ is Nash stable, we deduce that each player is Nash stable. Since player b is Nash stable, we have

$$v_b(P_{i'} \cup \{b\}) \leq v_b(P_{j'}). \tag{4.2}$$

Since (b, a) belongs to E, we deduce that $v_b(a) = 1$ and hence

$$v_b(P_{i'} \cup \{b\}) = \frac{\sum_{w\in P_{i'}\cup\{b\}} v_b(w)}{|P_{i'}|+1} \geq \frac{v_b(a)}{|P_{i'}|+1} = \frac{1}{|P_{i'}|+1}.$$

Furthermore, since the out-degree of player b is bounded by Δ and a is an out-neighbor of b outside $P_{j'}$, we deduce that b has at most $\Delta - 1$ out-neighbors in $P_{j'}$, that is, $\sum_{w\in P_{j'}} v_b(w) \leq \Delta-1$. Thus, $v_b(P_w) \leq \frac{\Delta-1}{|P_{j'}|}$. Using inequality (4.2), we have $\frac{1}{|P_{i'}|+1} \leq \frac{\Delta-1}{|P_{j'}|}$. Rearranging, we obtain $|P_{j'}| \leq \Delta|P_{i'}| - |P_{i'}| + \Delta - 1$. Furthermore, we deduce from $Q(k-j')$ and $n \geq k\Delta^{k+1}$ that

$$|P_{j'}| \geq \frac{n}{k \cdot \Delta^{k-j'}} \geq \frac{k\Delta^{k+1}}{k \cdot \Delta^{k-j'}} = \Delta^{j'+1}.$$

Notice that $j' \geq k - i + 1 \geq 2$ as $i \leq k - 1$. Therefore, we have $\Delta^3 \leq \Delta^{j'+1} \leq \Delta|P_{i'}| - |P_{i'}| + \Delta - 1$.

We claim that $|P_{i'}| \geq \Delta$. Assume for the sake of contradiction that $|P_{i'}| < \Delta$. Hence $\Delta|P_{i'}| - |P_{i'}| + \Delta - 1 \leq \Delta^2 + \Delta - 1$. Since $\Delta \geq 2$, we deduce

that $\Delta^2+\Delta-1 < 2\Delta^2 \leq \Delta^3$, a contradiction since $\Delta^3 \leq \Delta|P_{i'}|-|P_{i'}|+\Delta-1$. Hence $|P_{i'}| \geq \Delta$.

Since $|P_{j'}| \leq \Delta|P_{i'}| - |P_{i'}| + \Delta - 1$ and $|P_{i'}| \geq \Delta$, we have

$$|P_{j'}| \leq \Delta|P_{i'}| - \Delta + \Delta - 1 < \Delta|P_{i'}|.$$

Hence $|P_{i'}| > \frac{1}{\Delta}|P_{j'}|$, as required. □

4.4.1 Hardness

In this section, for each $k \geq 2$, we establish the NP-completeness of determining whether a directed weighted graph with edges weights -1 admits a Nash stable k-partition. We give an NP-completeness proof first for $k = 2$ and then for $k \geq 3$.

The following observation allows us to consider Nash stable partitions in FHGs played on undirected unweighted graphs with utility-minimizing players, rather than Nash stability in weighted directed graphs with negative edge weights and utility-maximizing players.

Observation 4.4.2. *Let $H = (N, E, v)$ denote a directed weighted graph where every edge has weight -1, let $H' = (N, E)$ denote the directed unweighted graph that contains the same vertices and edges as H, let k belong to $[|N|]$, and let $\mathcal{P}$ denote a k-partition in $\Pi_k(N)$. Then the k-partition $\mathcal{P}$ is Nash stable in $\mathcal{G}(H)$ with utility-maximizing players if and only if $\mathcal{P}$ is Nash stable in $\mathcal{G}(H')$ with utility-minimizing players.*

We now prove the following lemma, which will be used in our NP-completeness proofs for $k = 2$ and for $k \geq 3$.

Lemma 4.4.3. *Let N denote a set of utility-minimizing players, let $G = (N, E)$ be an unweighted directed graph, let k and i be integers such that $k \geq 2$ and i belongs to $[k-1]$, let x denote a player in N with exactly $k-1$ out-neighbors $y_1, \ldots, y_{k-1}$ in G, and let $\mathcal{P}$ denote a Nash stable k-partition in $\Pi_k(N)$. Then $\mathcal{P}(x) \neq \mathcal{P}(y_i)$.*

Proof. Assume for the sake of contradiction that $\mathcal{P}(x) = \mathcal{P}(y_i)$. Since y_i belongs to $\mathcal{P}(y_i)$ and y_i is an out-neighbor of x, we deduce that $v_x(\mathcal{P}(x)) = v_x(\mathcal{P}(y_i)) > 0$. Since x has exactly $k-1$ out-neighbors, there is a coalition P in the k-partition $\mathcal{P}$ that does not contain x or any out-neighbor of x. Therefore, $v_x(P \cup \{x\}) = 0 < v_x(\mathcal{P}(x))$. Hence the utility-minimizing player x is not Nash stable for $\mathcal{P}$, a contradiction. □

4.4.1.1 Nash Stable 2-Partitions

In this section, we use a reduction from the balanced unfriendly 2-partition problem to prove that it is NP-complete to determine whether a directed weighted graph where all edges have weights -1 admits a Nash stable 2-partition. A 2-partition of an undirected graph is called unfriendly if each vertex has at least as many neighbors outside its partition as it has inside its partition. Bazgan et al. [25] prove that determining whether a graph admits a balanced unfriendly 2-partition is NP-complete. Our reduction borrows

ideas from Kun et al. [113], who use an elegant reduction from the balanced unfriendly 2-partition problem to show that determining whether a directed graph has a stable coloring with two colors is NP-complete. They use a gadget that forces any stable 2-coloring to be balanced. We adapt this gadget to our setting to ensure that any Nash stable 2-partition is balanced.

Lemma 4.4.4. *For FHGs with utility-minimizing players and played on directed unweighted graphs, the Nash stable* 2*-partition problem is NP-complete.*

Proof. This problem is clearly in NP. For hardness, we reduce from the balanced unfriendly 2-partition problem. Let $G = (V, E)$ be an instance of the balanced unfriendly 2-partition problem, where G is an undirected graph and $|V|$ is even. We construct a directed graph $G' = (V', E')$ as follows.

Let n denote $|V|$ and let $M = (n+2)(n+5)$. Let graph G' consists of M isolated directed 2-cycles, four additional subgraphs, and some additional edges between these subgraphs. For any i in $[M]$, the vertices of the ith isolated directed 2-cycle are denoted a_i and b_i. The first subgraph is a "directed version" of G: it has vertex set V and two directed edges (u, v) and (v, u) for each undirected edge (u, v) in E. The second subgraph is a single vertex u_0. The third subgraph consists of two vertices v_0 and v_1, plus the directed edge (v_1, v_0). The fourth subgraph is a directed cycle with three vertices c_1, c_2, and c_3 and directed edges (c_1, c_2), (c_2, c_3), and (c_3, c_1). The following additional edges are included in G': a directed edge from u_0 to each vertex in $V + v_0$; a directed edge from v_0 to each vertex in $V + u_0$; (c_1, u_0); (c_1, v_1). Thus G' has

$n+1+2+3+2M = n+2M+6$ vertices and $2|E|+1+3+2(|N|+1)+2M = 2|E|+2M+2|N|+6$ directed edges.

We start by proving that if G has a balanced unfriendly partition (P_1, P_2) in $\Pi_2(V)$, then $\mathcal{G}(G')$ admits a Nash stable 2-partition. Let P'_1 denote $P_1 \cup \{u_0\} \cup \{v_1\} \cup \{c_3\} \cup \{a_i \mid i \in [M]\}$ and let P'_2 denote $P_2 \cup \{v_0\} \cup \{c_1, c_2\} \cup \{b_i \mid i \in [M]\}$. Clearly, (P'_1, P'_2) belongs to $\Pi_2(V')$ and $|P'_1| = |P'_2| = n/2 + M + 2 = |V'|/2$. It is straightforward to prove that the partition (P'_1, P'_2) is Nash stable by verifying that each player has at least as many out-neighbors outside its partition as it has inside its partition.

It remains to prove that if $\mathcal{G}(G')$ admits a Nash stable 2-partition, then G has a balanced unfriendly 2-partition. Let $\mathcal{P}' = (P'_1, P'_2)$ be a Nash stable 2-partition of $\mathcal{G}(G')$. We will prove that $(P'_1 \cap V, P'_2 \cap V)$ is a balanced unfriendly 2-partition of G. To do so, we first prove that $|P'_1 \cap V| = |P'_2 \cap V| = n/2$, i.e, that $(P'_1 \cap V, P'_2 \cap V)$ is balanced. We then proceed to prove that $|P'_1| = |P'_2|$. Once we have established that $|P'_1| = |P'_2|$, it is straightforward to deduce that for each i in $\{1, 2\}$, a utility-minimizing player v in $P'_i \cap V$ is Nash stable if and only if the corresponding vertex v in G has at least as many neighbors in $P'_{3-i} \cap V$ as in $P'_i \cap V$. Hence the balanced 2-partition $(P'_1 \cap V, P'_2 \cap V)$ is unfriendly, which concludes the proof of this lemma.

Notice that the players in $\{a_i, b_i \mid i \in [M]\}$ each have exactly one out-neighbor. By Lemma 4.4.3, we deduce that $\mathcal{P}'(a_i) \neq \mathcal{P}'(b_i)$ for each i in $[M]$. Since $\mathcal{P}' = (P'_1, P'_2)$ is a 2-partition, we conclude that $|P'_1 \cap \{a_i, b_i\}| = 1$ for each i in M. Thus $|P'_1| \geq M$ and $|P'_2| \geq M$.

We now prove the following useful claim.

Claim 1: $\mathcal{P}(c_3) \neq \mathcal{P}(c_2)$ and $\mathcal{P}(c_3) \neq \mathcal{P}(c_1)$.

Proof: Since c_3 is the unique out-neighbor of c_2 and c_1 is the unique out-neighbor of c_3, we deduce by Lemma 4.4.3 that $\mathcal{P}(c_3) \neq \mathcal{P}(c_2)$ and $\mathcal{P}(c_3) \neq \mathcal{P}(c_1)$. This completes the proof of Claim 1.

We now prove that $|P_1' \cap V| = |P_2' \cap V|$. Assume for the sake of contradiction that $|P_1' \cap V| \neq |P_2' \cap V|$. Without loss of generality, assume that $|P_1' \cap V| > |P_2' \cap V|$. Therefore, $|P_1' \cap V| \geq |P_2' \cap V| + 2$ as $|P_1' \cap V| + |P_2' \cap V| = |V|$ and $|V|$ is even. Claim 3 below contradicts the Nash stability of $\mathcal{P}$, allowing us to conclude that $|P_1' \cap V| = |P_2' \cap V|$. Before stating Claim 3, we establish Claim 2 below. Claims 1 and 2 are used in the proof Claim 3.

Claim 2: Vertex u_0 belongs to P_2' and vertex v_1 belongs to P_1'.

Proof: We first prove that u_0 belongs to P_2'. Assume for the sake of contradiction that u_0 belongs to P_1'. It suffices to prove that u_0 is not Nash stable, which yields a contradiction. Since all players in $P_1' \cap V$ are out-neighbors of u_0, we deduce that the utility of u_0 is at least

$$\frac{|P_1' \cap V|}{|P_1'|} \geq \frac{|P_2' \cap V| + 2}{|P_1'|} = \frac{|P_2' \cap V| + 2}{|V'| - |P_2'|} \geq \frac{|P_2' \cap V| + 2}{n + M + 6},$$

where the last inequality follows from $|P_2'| \geq M$ and $|V'| = n + 2M + 6$. Since there are $|V'| + 1$ out-neighbors of u_0 in G' and all players in $P_1' \cap V$ are out-neighbors of u_0, player u_0 has at most $|V'| + 1 - |P_1' \cap V|$ out-neighbors in P_2'.

Therefore, by deviating to P_2', player u_0 achieves utility at most

$$\frac{|V'|+1-|P_1'\cap V|}{|P_2'|+1}=\frac{|P_2'\cap V|+1}{|P_2'|+1}\leq\frac{|P_2'\cap V|+1}{M+1},$$

where the last inequality follows from $|P_2'|\geq M$. We now prove that the utility of player u_0 decreases if u_0 deviates to P_2', i.e., $\frac{|P_2'\cap V|+1}{M+1}<\frac{|P_2'\cap V|+2}{n+M+6}$. This is equivalent to show that $\frac{n+M+6}{M+1}<\frac{|P_2'\cap V|+2}{|P_2'\cap V|+1}$. Subtracting 1 from both sides, the desired inequality becomes $\frac{n+5}{M+1}<\frac{1}{|P_2'\cap V|+1}$. Since $M=(n+2)(n+5)$, we deduce that $\frac{n+5}{M+1}<\frac{n+5}{M}=\frac{n+5}{(n+5)(n+2)}=\frac{1}{n+2}$. Furthermore, since $|P_2'\cap V|\leq|V|=n<n+1$, we deduce that $\frac{1}{n+2}\leq\frac{1}{|P_2'\cap V|+1}$. Thus we conclude that $\frac{n+5}{M+1}<\frac{1}{n+2}\leq\frac{1}{|P_2'\cap V|+1}$, as desired. Hence the utility of player u_0 decreases if u_0 deviates to P_2'. It follows that player u_0 is not Nash stable, a contradiction. Therefore, we conclude that u_0 belongs to P_2'. A similar argument can be used to prove that v_0 is in P_2'. Since v_1 has exactly one out-neighbor v_0, we deduce by Lemma 4.4.3 that $\mathcal{P}'(v_1)\neq\mathcal{P}'(v_0)=P_2'$, i.e., $\mathcal{P}'(v_1)=P_1'$. This completes the proof of Claim 2.

Claim 3: At least one player in $\{c_1,c_2,c_3\}$ is not Nash stable.

Proof: Let i be an integer in $\{1,2\}$ such that $P_i'=\mathcal{P}(c_3)$. Claim 1 implies that players c_1 and c_2 belong to P_{3-i}'. It suffices to prove that c_1 is not Nash stable. Note that P_{3-i}' contains exactly one player in $\{u_0,v_1\}$ as u_0 is in P_2' and v_1 is in P_1' by Claim 2. Therefore, player c_1 has exactly two out-neighbors in P_{3-i}' and hence has utility $2/|P_{3-i}'|$. If c_1 deviates to P_i', then c_1 achieves utility $1/(|P_i'|+1)$. Notice that $2/|P_{3-i}'|>1/(|P_i'|+1)$ since $2(|P_i'|+1)\geq 2(M+1)>n+M+6=|V'|-M\geq|V'|-|P_i'|=|P_{3-i}'|$.

Therefore, the utility of player c_1 decreases if c_1 deviates to P'_i, and hence c_1 is not Nash stable. This completes the proof of Claim 3.

We now seek to prove that $|P'_1| = |P'_2|$. Note that we have previously deduced that $|P'_1 \cap \{a_i, b_i \mid i \in [M]\}| = |P'_2 \cap \{a_i, b_i \mid i \in [M]\}| = M$, $|P'_1 \cap V| = |P'_2 \cap V|$, and $|P'_1 \cap \{v_0, v_1\}| = |P'_2 \cap \{v_0, v_1\}| = 1$. Thus, it remains to prove that $|P'_1 \cap \{u_0, c_1, c_2, c_3\}| = |P'_2 \cap \{u_0, c_1, c_2, c_3\}|$. Let i be an integer such that c_3 belongs to P'_1. Thus Claim 1 implies that c_1 and c_2 belong to P'_{3-i}. It suffices to show that u_0 belongs to P'_i. Assume for the sake of contradiction that u_0 belongs to P'_{3-i}. Therefore, coalition P'_{3-i} contains at least two of c_1's three out-neighbors (i.e., u_0 and c_2). Thus, c_1 is not Nash stable, a contradiction. Therefore, we conclude that u_0 is in P'_1, as required. □

4.4.1.2 Nash Stable k-Partitions

Lemma 4.4.5. *For FHGs on directed unweighted graphs with utility-minimizing players, the Nash stable k-partition problem is NP-complete for all $k \geq 3$.*

Proof. Clearly, this problem is in NP. For hardness, we reduce from the problem shown to be complete in Lemma 4.4.4. The proof of Lemma 4.4.4 shows that the latter problem remains NP-complete if we restrict the attention to instances in which the associated graph contains an isolated 2-cycle. Let $G = (V, E)$ denote such an instance that contains an isolated 2-cycle with vertices p_1 and p_2. Let n denote $|V|$, let M denote $n^2 - 2n + 2$, and suppose that $n \geq 3$. We construct a directed unweighted graph $G' = (V', E')$ as

follows.

We define V' as

$$V \cup \{p_3, \ldots, p_k\} \cup \{d_{j,\ell} \mid j \in \{3, \ldots, k\},\ \ell \in [M]\}.$$

Recall that $\{p_1, p_2\} \subseteq V$ and $\{(p_1, p_2), (p_2, p_1)\} \subseteq E$. We define E' as $E \cup E_1 \cup E_2$, where E_1 and E_2 are defined as follows. The set E_1 consists of all edges associated with a directed clique over $\{p_1, \ldots, p_k\}$. (Thus $E \cap E_1 = \{(p_1, p_2), (p_2, p_1)\}$.) For any j and ℓ such that $3 \leq j \leq k$ and ℓ belongs to $[M]$, the set E_2 includes an edge from $d_{j,\ell}$ to each player in $\{p_1, \ldots, p_k\} \setminus \{p_j\}$ and an edge from each vertex in $V \setminus \{p_1, p_2\}$ to $d_{j,\ell}$.

Claims 1 and 2 below imply that the lemma holds

Claim 1: If $\mathcal{G}(G)$ admits a Nash stable 2-partition $\mathcal{P} = (P_1, P_2)$ for utility-minimizing players, then $\mathcal{G}(G')$ admits a Nash stable k-partition for utility-minimizing players.

Let $\mathcal{P}'$ denote the k-partition $(P'_i)_{i \in [k]}$, where $P'_1 = P_1$, $P'_2 = P_2$, and $P'_i = \{p_i\} \cup \{d_{i,\ell} \mid \ell \in [M]\}$ for each i in $\{3, \ldots, k\}$. Clearly, for each i in $\{3, \ldots, k\}$, all players in $P'_i = \{p_i\} \cup \{d_{i,\ell} \mid \ell \in [M]\}$ have utility 0, and hence are Nash stable. To prove that $\mathcal{P}'$ is Nash stable, it remains to prove that all of the players in $P'_1 \cup P'_2$ are Nash stable. Assume for the sake of contradiction that there is an integer i in $\{1, 2\}$ and player p in P'_i such that p is not Nash stable. Thus, there is a coalition P'_j such that the utility of player p decreases if p deviates to P'_j. Since $P'_1 = P_1$, $P'_2 = P_2$, and the 2-partition (P_1, P_2) is Nash stable, we deduce that $j \neq 3 - i$. Therefore, j belongs to $\{3, \ldots, k\}$. Notice

that for each ℓ in $[M]$, the player $d_{j,q}$ is an out-neighbor of p in P'_j. Therefore, $v_p(P'_j \cup \{p\}) = M/(M+2) = 1 - 2/(M+2)$. Moreover, p has at most $|P'_i| - 1$ out-neighbors in P'_i and hence $v_p(P'_j \cup \{p\}) \leq 1 - 1/|P'_i| \leq 1 - 1/(n-1)$ as $|P'_i| = |P_i| = |V \setminus P_{3-i}| \leq n - 1$. (The latter inequality holds since P_1 and P_2 are each required to be non-empty.) Note that $M = n^2 - 2n + 2 = (n-1)^2 + 1 > 2n - 2$ as $n \geq 3$. Therefore, we have $2/(M+2) < 1/n$, i.e., $v_p(P'_j \cup \{p\}) = 1 - 2/(M+2) > 1 - 1/n \geq v_p(P'_i)$. Thus the utility of player p does not decrease if p deviates to P'_j, a contradiction. This completes the proof of Claim 1.

Claim 2: If $\mathcal{G}(G')$ admits a Nash stable k-partition $\mathcal{P}' = (P'_i)_{i \in [k]}$ in $\Pi_k(V')$ for utility-minimizing players, then $\mathcal{G}(G)$ admits a Nash stable 2-partition for utility-minimizing players.

Notice that for each i in $[k]$, the player p_i has $k-1$ out-neighbors in $\{p_j \mid j \in [k] \setminus \{i\}\}$. Thus Lemma 4.4.3 implies that $\mathcal{P}'(p_i) \neq \mathcal{P}'(p_j)$ for all j in $[k] \setminus \{i\}$. Therefore, the k vertices $p_1, \ldots, p_k$ belong to distinct coalitions of the k-partition. Without loss of generality, suppose that P'_i contains p_i for all i in $[k]$. Hence $\mathcal{P}'(p_i) = P'_i$ for all i in $[k]$. It now suffices to prove that (P'_1, P'_2) is a 2-partition in $\Pi_2(V)$. After we prove this, the desired statement that $\mathcal{G}(G)$ admits the Nash stable 2-partition (P'_1, P'_2) directly follows, since the Nash stability of (P'_1, P'_2) follows from the Nash stability of $\mathcal{P}' = (P'_i)_{i \in [k]}$.

To prove that (P'_1, P'_2) is in $\Pi_2(V)$, it suffices to prove that $P'_1 \cup P'_2 = V$. Let $Q = \cup_{3 \leq i \leq k} P'_i$. Below we prove that $P'_1 \cup P'_2 = V$ by showing that $V' \setminus V \subseteq Q$ and $V \cap Q = \emptyset$ hold.

We first prove that $V' \setminus V \subseteq Q$. Since p_i belongs to P'_i for each i in $\{3, \ldots, k\}$, it remains to consider the dummy vertices. For each j in $\{3, \ldots, k\}$ and any ℓ in $[M]$, the dummy vertex $d_{j,\ell}$ has $k-1$ out-neighbors in $\{p_i \mid i \in [k] \setminus \{j\}\}$, and hence we deduce by Lemma 4.4.3 that $\mathcal{P}'(d_{j,\ell}) \neq \mathcal{P}'(p_i) = P'_i$ for all i in $[k] \setminus \{j\}$. Hence we conclude that $\mathcal{P}'(d_{j,\ell}) = P'_j$ for all i in $[k] \setminus \{j\}$. Therefore, for all j in $\{3, \ldots, k\}$, we deduce that $\{p_j\} \cup \{d_{j,\ell} \mid \ell \in [M]\} \subseteq P'_j$. In summary, we have determined the coalition that contains each vertex in the set

$$\{p_1, \ldots, p_k\} \cup \{d_{j,\ell} \mid j \in \{3, \ldots, k\},\ \ell \in [M]\}.$$

Of these vertices, one belongs to P'_1, one belongs to P'_2, and $M+1$ vertices belong to P'_j for each j in $\{3, \ldots, k\}$. Since the number of other vertices in V' is $|V \setminus \{p_1, p_2\}| = n-2$, we conclude that $|P'_1| \leq n-1$, $|P'_2| \leq n-1$, and $|P'_j| \leq 1 + M + n - 2 = M + n - 1$ for all j in $\{3, \ldots, k\}$.

Now we prove that $V \cap Q = \emptyset$. Since p_1 belongs to P'_1 and p_2 belongs to P'_2 , it is sufficient to prove that $(V \setminus \{p_1, p_2\}) \cap Q = \emptyset$. Assume for the sake of contradiction that there is a player p in $V \setminus \{p_1, p_2\}$ such that p does not belong to $P'_1 \cup P'_2$. Let j be an integer in $\{3, \ldots, k\}$ such that $P'_j = \mathcal{P}'(p)$. Note that P'_j contains $\{d_{j,\ell} \mid \ell \in [M]\}$, and hence P'_j contains at least M out-neighbors of p. Thus $v_p(P'_j) \geq \frac{M}{|P'_j|} \geq \frac{M}{M+n-1} = 1 - \frac{n-1}{M+n-1}$, where the last inequality follows from $|P'_j| \leq M + n - 1$. Moreover, $v_p(P'_1 \cup \{p\}) \leq \frac{|P'_1 \cup \{p\}|-1}{|P'_1 \cup \{p\}|} \leq \frac{n-1}{n} = 1 - \frac{1}{n}$, where the last inequality follows from $|P_1| \leq n-1$. Since $M = n^2 - 2n + 2$, it follows that $\frac{1}{n} = \frac{n-1}{n(n-1)} > \frac{n-1}{n^2-n+1} = \frac{n-1}{M+n-1}$, i.e., $v_p(P'_1 \cup \{p\}) < v_p(P'_j)$. Hence the utility of player p decreases if p deviates to P'_1, contradicting the Nash

stability of p. □

Lemmas 4.4.4 and 4.4.5 together yield Theorem 4.4.6 below, the main result of this section.

Theorem 4.4.6. *For FHGs played on directed weighted graphs where all edges have weight* -1*, the Nash stable* k*-partition problem is NP-complete for every fixed* $k \geq 2$*.*

Chapter 5

The Obnoxious Facility Location Game with Dichotomous Preferences

5.1 Introduction

The facility location game (FLG) was introduced by Procaccia and Tannenholtz [147]. In this setting, a central planner wants to build a facility that serves agents located on a path. The agents report their locations, which are fed to a mechanism that decides where the facility should be built. Procaccia and Tannenholtz studied two different objectives that the planner seeks to minimize: the sum of the distances from the facility to all agents and the maximum distance of any agent to the facility.

Every agent aims to maximize their welfare, which increases as their distance to the facility decreases. An agent or a coalition of agents can misreport their location(s) to try to increase their welfare. It is natural to seek strategyproof (SP) or group-strategyproof (GSP) mechanisms, which incentivize truthful reporting. Often such mechanisms cannot simultaneously opti-

The results presented in this chapter are based on Greg Plaxton and Vaibhav Sinha [117]. My main contribution to these results is to prove the existence of a mechanism that is WGSP and efficient for three facilities in the path setting. This dissertation does not include the details of our results related to egalitarian mechanisms, or to the cycle and square settings; these results can be found in Vaibhav Sinha's M.Sc. thesis [156].

mize the planner's objective. In these cases, it is desirable to approximately optimize the planner's objective.

In real scenarios, an agent might dislike a certain facility, such as a power plant, and want to stay away from it. This variant, called the obnoxious facility location game (OFLG), was introduced by Cheng et al., who studied the problem of building an obnoxious facility on a path [55]. In the present chapter, we consider the problem of building multiple obnoxious facilities on a path. With multiple facilities, there are different ways to define the welfare function. For example, in the case of two facilities, the welfare of the agent can be the sum, minimum, or maximum of the distances to the two facilities. In our work, as all the facilities are obnoxious, a natural choice for welfare is the minimum distance to any obnoxious facility: the closest facility to an agent causes them the most annoyance, and if it is far away, then the agent is satisfied.

A facility might not be universally obnoxious. Consider, for example, a school or sports stadium. An agent with no children might consider a school to be obnoxious due to the associated noise and traffic, while an agent with children might not consider it to be obnoxious. Another agent who is not interested in sports might similarly consider a stadium to be obnoxious. We assume that each agent has dichotomous preferences; they dislike some subset of the facilities and are indifferent to the others. Each agent reports a subset of facilities to the planner. As the dislikes are private information, the reported subset might not be the subset of facilities that the agent truly dislikes. On

the other hand, we assume that the agent locations are public and cannot be misreported.

In this chapter, we study a variant of FLG, which we call DOFLG (Dichotomous Obnoxious Facility Location Game), that combines the three aspects mentioned above: multiple (heterogeneous) obnoxious facilities, minimum distance as welfare, and dichotomous preferences. We seek to design mechanisms that perform well with respect to either a utilitarian or egalitarian objective. The utilitarian objective is to maximize the social welfare, that is, the total welfare of all the agents. A mechanism that maximizes social welfare is said to be efficient. The egalitarian objective is to maximize the minimum welfare of any agent. For both objectives, we seek mechanisms that are SP, or better yet, weakly or strongly group-strategyproof (WGSP / SGSP).

Our contributions. We study DOFLG with n agents. We consider the utilitarian objective. We present 2-approximate SGSP mechanisms for any number of facilities when the agents are located on a path, cycle, or square. We obtain the following two additional results for the path setting. In the first main result of the chpater, we obtain a mechanism that is WGSP for any number of facilities and efficient for up to three facilities. To show that this mechanism is WGSP, we relate it to a weighted approval voting mechanism. To prove its efficiency, we identify two crucial properties that the welfare function satisfies, and we use an exchange argument. For the path setting, we also show that no SGSP mechanism can achieve an approximation ratio better than 5/4, even for one facility.

Table 5.1: Summary of our results for DOFLG when the agents are located on a path.

	Utilitarian		Egalitarian	
	LB	UB	LB	UB
SP	**1**	1 for $k \leq 3$	**1**	**1**
WGSP			$\Omega(\sqrt{n})$	$O(n)$
SGSP	5/4	**2**		

We consider the egalitarian objective. We provide SP mechanisms for any number of facilities when the agents are located on a path, cycle, or square. In the second main result of the chapter, we prove that the approximation ratio achieved by any WGSP mechanism is $\Omega(\sqrt{n})$, even for two facilities. Also, we present a straightforward $O(n)$-approximate WGSP mechanism. Both of the results for WGSP mechanisms hold for DOFLG when the agents are located on a path or cycle. Table 5.1 summarizes our results. The heading LB (resp., UB) stands for lower (resp., upper) bound. The results in the egalitarian column also hold when the agents are located on a cycle. Boldface results hold when the agents are located on a path, cycle, or square.

My main contribution to these results is to prove the existence of a mechanism that is WGSP and efficient for three facilities in the path setting. This dissertation does not include the details of our results related to egalitarian mechanisms, or to the cycle and square settings; these results can be found in Vaibhav Sinha's M.Sc. thesis [156].

The rest of this chapter is organized as follows. Section 5.2 reviews related work. Section 5.3 introduces some basic definitions and notations.

Section 5.4 defines the weighted approval voting mechanism and establishes several useful properties of this mechanism. Section 5.5 presents our main results for the path setting.

5.2 Related Work

FLG was introduced by Procaccia and Tannenholtz [147]. Many generalizations and extensions of FLG have been studied [10, 65, 75, 76, 81, 82, 122, 169]; here we highlight some of the most relevant work. Cheng et al. introduced OFLG and presented a WGSP mechanism to build a single facility on a path [55]. Later they extended the model to cycles and trees [56]. A complete characterization of single-facility SP/WGSP mechanisms for paths has been developed [96]. Duan et al. studied the problem of locating two obnoxious facilities at least distance d apart [70]. Other variants of OFLG have been considered [54, 84, 138, 166].

Agent preferences over the facilities were introduced to FLG in [170]. Serafino and Ventre studied FLG for building two facilities where each agent likes a subset of the facilities [153]. Anastasiadis and Deligkas extended this model to allow the agents to like, dislike, or be indifferent to the facilities [12]. The aforementioned works address linear (sum) welfare function. Yuan et al. studied non-linear welfare functions (max and min) for building two non-obnoxious facilities [168]; their results have subsequently been strengthened [53, 121]. In the present chapter, we initiate the study of a non-linear welfare function (min) for building multiple obnoxious facilities.

5.3 Preliminaries

The problems considered in this chapter involve a set of agents located on a path, cycle, or square. In the path (resp., cycle, square) setting, we assume without loss of generality that the path (resp., cycle, square) is the unit interval (resp., unit-circumference circle, unit square). We map the points on the unit-circumference circle to $[0, 1)$, in the natural manner. Thus, in the path (resp., cycle, square) setting, each agent i is located in $[0, 1]$ (resp., $[0, 1)$, $[0, 1]^2$). The distance between any two points x and y is denoted $\Delta(x, y)$. In the path and square settings, $\Delta(x, y)$ is defined as the Euclidean distance between x and y. In the cycle setting, $\Delta(x, y)$, is defined as the length of the shorter arc between x and y. In all settings, we index the agents from 1. Each agent has a specific location in the path, cycle, or square. A *location profile* $\mathbf{x}$ is a vector $(x_1, \ldots, x_n)$ of points, where n denotes the number of agents and x_i is the location of agent i. Sections 5.5 presents our results for the path.

Consider a set of agents 1 through n and a set of facilities $\mathcal{F}$, where we assume that each agent dislikes (equally) certain facilities in $\mathcal{F}$ and is indifferent to the rest. In this context, we define an *aversion profile* $\mathbf{a}$ as a vector $(a_1, \ldots, a_n)$ where each component a_i is a subset of $\mathcal{F}$. We say that such an aversion profile is *true* if each component a_i is equal to the subset of $\mathcal{F}$ disliked by agent i. In this chapter, we also consider *reported* aversion profiles where each component a_i is equal to the set of facilities that agent i claims to dislike. Since agents can lie, a reported aversion profile need not be true. For any aversion profile $\mathbf{a}$ and any subset C of agents $[n]$, $\mathbf{a}_C$ (resp., $\mathbf{a}_{-C}$) denotes the

aversion profile for the agents in (resp., not in) C. For a singleton set of agents $\{i\}$, we abbreviate $\mathbf{a}_{-\{i\}}$ as $\mathbf{a}_{-i}$.

An instance of the dichotomous obnoxious facility location (DOFL) problem is given by a tuple $(n, k, \mathbf{x}, \mathbf{a})$ where n denotes the number of agents, there is a set of k facilities $\mathcal{F} = \{F_1, \ldots, F_k\}$ to be built, $\mathbf{x} = (x_1, \ldots, x_n)$ is a location profile for the agents, and $\mathbf{a} = (a_1, \ldots, a_n)$ is an aversion profile (true or reported) for the agents with respect to $\mathcal{F}$. A solution to such a DOFL instance is a vector $\mathbf{y} = (y_1, \ldots, y_k)$ where component y_j specifies the point at which to build F_j. We say that a DOFL instance is true (resp., reported) if the associated aversion profile is true (resp., reported). For any DOFL instance $I = (n, k, \mathbf{x}, \mathbf{a})$ and any j in $[k]$, we define haters(I, j) as $\{i \in [n] \mid F_j \in a_i\}$, and indiff$(I)$ as $\{i \in [n] \mid a_i = \emptyset\}$.

For any DOFL instance $I = (n, k, \mathbf{x}, \mathbf{a})$ and any associated solution $\mathbf{y}$, we define the *welfare* of agent i, denoted $w(I, i, \mathbf{y})$, as $\min_{j:F_j \in a_i} \Delta(x_i, y_j)$, i.e., the minimum distance from x_i to any facility in a_i. Remark: If a_i is empty, we define $w(I, i, \mathbf{y})$ as $1/2$ in the cycle setting, $\max(\Delta(x_i, 0), \Delta(x_i, 1))$ in the path setting, and the maximum distance from x_i to a corner in the square setting.

The foregoing definition of agent welfare is suitable for true DOFL instances, and is only meaningful for reported DOFL instances where the associated aversion profile is close to true. In this chapter, reported aversion profiles arise in the context of mechanisms that incentivize truthful reporting, so it is reasonable to expect such aversion profiles to be close to true. We define the *social welfare* (resp., *minimum welfare*) as the sum (resp., minimum) of the

individual agent welfares. When the facilities are built at $\mathbf{y}$, the social welfare and minimum welfare are denoted by $\mathrm{SW}(I, \mathbf{y})$ and $\mathrm{MW}(I, \mathbf{y})$, respectively. Thus $\mathrm{SW}(I, \mathbf{y}) = \sum_{i \in [n]} w(I, i, \mathbf{y})$ and $\mathrm{MW}(I, \mathbf{y}) = \min_{i \in [n]} w(I, i, \mathbf{y})$.

Definition 5.3.1. *For $\alpha \geq 1$, a DOFL algorithm A is α-efficient if for any DOFL instance I,*

$$\max_{\mathbf{y}} \mathrm{SW}(I, \mathbf{y}) \leq \alpha \, \mathrm{SW}(I, A(I)).$$

Similarly, A is α-egalitarian if for any DOFL instance I,

$$\max_{\mathbf{y}} \mathrm{MW}(I, \mathbf{y}) \leq \alpha \, \mathrm{MW}(I, A(I)).$$

A 1-efficient (resp., 1-egalitarian) DOFL algorithm, is said to be efficient (resp., egalitarian).

We are now ready to define a DOFL-related game, which we call DOFLG. It is convenient to describe a DOFLG instance in terms of a pair (I, I') of DOFL instances where $I = (n, k, \mathbf{x}, \mathbf{a})$ is true and $I' = (n, k, \mathbf{x}, \mathbf{a}')$ is reported. There are n agents indexed from 1 to n, and a planner. There is a set of k facilities $\mathcal{F} = \{F_1, \ldots, F_k\}$ to be built. The numbers n and k are publicly known, as is the location profile $\mathbf{x}$ of the agents. Each component a_i of the true aversion profile $\mathbf{a}$ is known only to agent i. Each agent i submits component a'_i of the reported aversion profile $\mathbf{a}'$ to the planner. The planner, who does not have access to $\mathbf{a}$, runs a DOFL algorithm, call it A, to map I' to a solution. The input-output behavior of A defines a DOFLG mechanism, call it M; in the special case where $k = 1$, we say that M is a single-facility

DOFLG mechanism. We would like to choose A so that M enjoys strong game-theoretic properties. We say that M is α-efficient (resp., α-egalitarian, efficient, egalitarian) if A is α-efficient (resp., α-egalitarian, efficient, egalitarian). As indicated earlier, such properties (which depend on the notion of agent welfare) are only meaningful if the reported aversion profile is close to true. To encourage truthful reporting, we require our mechanisms to be SP, as defined below; we also consider the stronger properties WGSP and SGSP.

The SP property says that no agent can increase their welfare by lying about their dislikes.

Definition 5.3.2. *A DOFLG mechanism M is SP if for any DOFLG instance (I, I') with $I = (n, k, \mathbf{x}, \mathbf{a})$, and $I' = (n, k, \mathbf{x}, \mathbf{a}')$, and any agent i in $[n]$ such that $\mathbf{a}' = (\mathbf{a}_{-i}, a'_i)$, we have*

$$w(I, i, M(I)) \geq w(I, i, M(I')).$$

The WGSP property says that if a non-empty coalition $C \subseteq [n]$ of agents lies, then at least one agent in C does not increase their welfare.

Definition 5.3.3. *A DOFLG mechanism M is WGSP if for any DOFLG instance (I, I') with $I = (n, k, \mathbf{x}, \mathbf{a})$, and $I' = (n, k, \mathbf{x}, \mathbf{a}')$, and any non-empty coalition $C \subseteq [n]$ such that $\mathbf{a}' = (\mathbf{a}_{-C}, \mathbf{a}'_C)$, there exists an agent i in C such that*

$$w(I, i, M(I)) \geq w(I, i, M(I')).$$

The SGSP property says that if a coalition $C \subseteq [n]$ of agents lies and some agent in C increases their welfare then some agent in C decreases their welfare.

Definition 5.3.4. *A DOFLG mechanism M is SGSP if for any DOFLG instance (I, I') with $I = (n, k, \mathbf{x}, \mathbf{a})$, and $I' = (n, k, \mathbf{x}, \mathbf{a}')$, and any coalition $C \subseteq [n]$ such that $\mathbf{a}' = (\mathbf{a}_{-C}, \mathbf{a}'_C)$, if there exists an agent i in C such that*

$$w(I, i, M(I)) < w(I, i, M(I')),$$

then there exists an agent i' in C such that

$$w(I, i', M(I)) > w(I, i', M(I')).$$

Every SGSP mechanism is WGSP and every WGSP mechanism is SP.

5.4 Weighted Approval Voting

Before studying efficient mechanisms for our problem, we review a variant of the approval voting mechanism [41]. An instance of Dichotomous Voting (DV) is a tuple $(m, n, \mathbf{C}, \mathbf{w}^+, \mathbf{w}^-)$ where m voters $1, \ldots, m$ have to elect a candidate among the set of candidates $C = \{c_1, \ldots, c_n\}$. Each voter i has dichotomous preferences, that is, voter i partitions all of the candidates into two equivalence classes: a top (most preferred) tier C_i and a bottom tier $\overline{C_i} = C \setminus C_i$. Each voter i has associated (and publicly known) weights $w_i^+ \geq w_i^- \geq 0$. The symbols $\mathbf{C}$, $\mathbf{w}^+$, and $\mathbf{w}^-$ denote length-m vectors with ith element C_i, w_i^+, and

w_i^-, respectively. We now present our weighted approval voting mechanism.

Mechanism 1. Given a DV instance $(m, n, \mathbf{C}, \mathbf{w}^+, \mathbf{w}^-)$, every voter i votes by partitioning C into C_i' and $\overline{C_i'}$. Let the weight function w be such that for voter i and candidate c_j, $w(i,j) = w_i^+$ if c_j is in C_i' and $w(i,j) = w_i^-$ otherwise. For any j in $[n]$, we define $A(j) = \sum_{i \in [m]} w(i,j)$ as the approval of candidate c_j. The candidate c_j with highest approval $A(j)$ is declared the winner. Ties are broken according to a fixed ordering of the candidates (e.g., in favor of lower indices).

We note that the approval voting mechanism can be obtained from the weighted approval voting mechanism by setting weights w_i^+ to 1 and w_i^- to 0 for all voters i. In Section 5.3, we defined SP, WGSP, and SGSP in the DOFLG setting. These definitions are easily generalized to the voting setting. Brams and Fishburn proved that the approval voting mechanism is SP [41]. Below we prove that our weighted approval voting mechanism is WGSP (and hence also SP).

Theorem 5.4.1. *Mechanism 1 is WGSP.*

Proof. Assume for the sake of contradiction that there is an instance in which a coalition of voters U with true preferences $\{(C_i, \overline{C_i})\}_{i \in U}$ all benefit by misreporting their preferences as $\{(C_i', \overline{C_i'})\}_{i \in U}$. For any candidate c_j, let $A(j)$

Our mechanism differs from the homonymous mechanism of Massó et al., which has weights for the candidates instead of the voters [164].

denote the approval of c_j when coalition U reports truthfully, and let $A'(j)$ denote the approval of c_j when coalition U misreports.

Let c_k be the winning candidate when coalition U reports truthfully, and let c_ℓ be the winning candidate when coalition U misreports. Since every voter in U benefits when the coalition misreports, we know that c_k belongs to $\bigcap_{i \in U} \overline{C_i}$ and c_ℓ belongs to $\bigcap_{i \in U} C_i$.

Since c_k belongs to $\bigcap_{i \in U} \overline{C_i}$, we deduce that $A'(k) = A(k) + \sum_{i \in U : c_k \in C'_i} w_i^+ - w_i^-$ and hence $A'(k) \geq A(k)$. Similarly, since c_ℓ belongs to $\bigcap_{i \in U} C_i$, we deduce that $A'(\ell) = A(\ell) + \sum_{i \in U : c_\ell \in \overline{C'_i}} w_i^- - w_i^+$ and hence $A(\ell) \geq A(\ell')$.

Since c_k wins when coalition U truthfully, one of the following two cases is applicable.

Case 1: $A(k) > A(\ell)$. Since $A'(k) \geq A(k)$ and $A(\ell) \geq A'(\ell)$, the case condition implies that $A'(k) > A'(\ell)$. Hence c_ℓ does not win when coalition U misreports, a contradiction.

Case 2: $A(k) = A(\ell)$ and c_k has higher priority than c_ℓ. Since $A'(k) \geq A(k)$ and $A(\ell) \geq A(\ell')$, the case condition implies that $A'(k) \geq A(\ell')$ and c_k has higher priority than c_ℓ. Hence c_ℓ does not win when coalition U misreports, a contradiction. □

5.5 Efficient Mechanisms for Unit Intervals

We now present our efficient mechanism for DOFLG.

Mechanism 2. For a given reported DOFL instance $I = (n, k, \mathbf{x}, \mathbf{a})$, output the

lexicographically least solution $\mathbf{y}$ in $\{0,1\}^k$ that maximizes the social welfare $\mathrm{SW}(I, \mathbf{y})$.

Theorem 5.5.1. *Mechanism 2 is WGSP.*

Proof. To establish this theorem, we show that Mechanism 2 can be equivalently expressed in terms of the approval voting mechanism. Hence Theorem 5.4.1 implies the theorem.

Let (I, I') denote a DOFLG instance where $I = (n, k, \mathbf{x}, \mathbf{a})$ and $I' = (n, k, \mathbf{x}, \mathbf{a}')$. We view each agent $i \in [n]$ as a voter, and each $\mathbf{y}$ in $\{0,1\}^k$ as a candidate. We obtain the top-tier candidates C_i of voter i, and their reported top-tier candidates C'_i, from a_i and a'_i, respectively. Assume without loss of generality that $x_i \leq 1/2$ (the other case can be handled similarly). Set $C_i = \{\mathbf{y} = (y_1, \ldots, y_k) \in \{0,1\}^k \mid y_j = 1 \text{ for all } F_j \in a_i\}$ and similarly $C'_i = \{\mathbf{y} = (y_1, \ldots, y_k) \in \{0,1\}^k \mid y_j = 1 \text{ for all } F_j \in a'_i\}$. Also set $w_i^+ = 1 - x_i$ and $w_i^- = x_i$. With this notation, it is easy to see that $A(\mathbf{y}) = \mathrm{SW}(I', \mathbf{y})$, and that choosing the $\mathbf{y}$ with the highest social welfare in Mechanism 2 is the same as electing the candidate with the highest approval in Mechanism 1. □

We show that Mechanism 2 is efficient for $k = 3$. First, we note a well-known result about the 1-Maxian problem. In this problem, there are n points located at $z_1, \ldots, z_n$ in the interval $[a, b]$, and the task is to choose a point in $[a, b]$ such that the sum of the distances from that point to all z_is is maximized.

Lemma 5.5.2 (Optimality of the 1-Maxian Problem)**.** *Let $[a, b]$ be a real interval, let $z_1, \ldots, z_n$ belong to $[a, b]$, and let $f(z)$ denote $\sum_{i\in[n]} |z - z_i|$. Then $\max_{z\in[a,b]} f(z)$ belongs to $\{f(a), f(b)\}$.*

Before proving the main theorem, we establish Lemma 5.5.3, which follows from Lemma 5.5.2.

Lemma 5.5.3. *Let $I = (n, k, \mathbf{x}, \mathbf{a})$ denote the reported DOFL instance, let Y denote the set of all y in $[0, 1]$ such that it is efficient to build all k facilities at y, and assume that Y is non-empty. Then $Y \cap \{0, 1\}$ is non-empty.*

Proof. Let U denote $\text{indiff}(I)$. When all of the facilities are built at y,

$$\text{SW}(I, (y, \ldots, y)) = \sum_{i\in[n]\setminus U} |x_i - y| + \sum_{i\in U} w(I, i, y).$$

Since Y is non-empty, $\max_y \text{SW}(I, (y, \ldots, y)) = \max_{\mathbf{y}} \text{SW}(I, \mathbf{y})$. Moreover, since $\sum_{i\in U} w(I, i, y)$ does not depend on y, Lemma 5.5.2 implies that

$$\max(\text{SW}(I, (0, \ldots, 0)), \text{SW}(I, (1, \ldots, 1))) = \max_y \text{SW}(I, (y, \ldots, y)).$$

Thus, if $\text{SW}(I, (0, \ldots, 0)) \geq \text{SW}(I, (1, \ldots, 1))$, it is efficient to build all k facilities at 0. Otherwise, it is efficient to build all k facilities at 1. □

Theorem 5.5.4. *Mechanism 2 is efficient for $k = 3$.*

Proof. Let $I = (n, k, \mathbf{x}, \mathbf{a})$ denote the reported DOFL instance and let $\mathbf{y}^* = (y_1^*, y_2^*, y_3^*)$ be an efficient solution for I such that $y_1^* \leq y_2^* \leq y_3^*$.

Consider fixing variables y_1 and y_3 in the social welfare function $\mathrm{SW}(I, \mathbf{y})$. That is, we have

$$\mathrm{SW}(I, \mathbf{y})|_{y_1=y_1^*, y_3=y_3^*} = \sum_{i\in[n]} w(I, i, \mathbf{y})|_{y_1=y_1^*, y_3=y_3^*}.$$

For convenience, let $\mathrm{SW}(y_2)$ denote $\mathrm{SW}(I, \mathbf{y})|_{y_1=y_1^*, y_3=y_3^*}$ and let $w_i(y_2)$ denote $w(I, i, \mathbf{y})|_{y_1=y_1^*, y_3=y_3^*}$ for each agent i.

Claim 1: For each agent i, the welfare function $w_i(y_2)$ with $y_2 \in [y_1^*, y_3^*]$ satisfies at least one of the following two properties:

1. $w_i(y_2) = |y_2 - x_i|$;
2. $w_i(y_1^*) = w_i(y_3^*) = \max_{y\in[y_1^*, y_3^*]} w_i(y)$.

Proof: Consider an agent i. We consider five cases.

Case 1: $F_2 \notin a_i$. Since the welfare of agent i is independent of the location of F_2, w_i is a constant function. Hence property 2 is satisfied.

Case 2: $a_i = \{F_2\}$. By definition, we have $w_i(y_2) = |y_2 - x_i|$. Hence property 1 is satisfied.

Case 3: $a_i = \{F_1, F_2\}$. By definition, we have $w_i(y_2) = \min(|y_1^* - x_i|, |y_2 - x_i|)$. Notice that $w_i(y_1^*) = \min(|y_1^* - x_i|, |y_1^* - x_i|) = |y_1^* - x_i| = \max_{y\in[y_1^*, y_3^*]} w_i(y)$. Moreover, $w_i(y_3^*) = \min(|y_1^* - x_i|, |y_3^* - x_i|)$. We consider two cases.

Case 3.1: $|y_1^* - x_i| > |y_3^* - x_i|$. Then $w_i(y_3^*) = |y_3^* - x_i|$ and hence $w_i(y_2) = |y_2 - x_i|$ for all y_2 in $[y_1^*, y_3^*]$, that is, $w_i(y_2)$ satisfies property 1.

Case 3.2: $|y_1^* - x_i| \leq |y_3^* - x_i|$. Then $w_i(y_3^*) = |y_1^* - x_i| = \max_{y \in [y_1^*, y_3^*]} w_i(y) = w_i(y_1^*)$ and hence $w_i(y_2)$ satisfies property 2.

Case 4: $a_i = \{F_2, F_3\}$. This case is symmetric to Case 3 and can be handled similarly.

Case 5: $a_i = \{F_1, F_2, F_3\}$. By definition, we have $w_i(y_2) = \min(|y_1^* - x_i|, |y_2 - x_i|, |y_3^* - x_i|)$. Notice that $w_i(y_1^*) = w_i(y_3^*) = \min(|y_1^* - x_i|, |y_3^* - x_i|)$. Also notice that for any y_2 in $[y_1^*, y_3^*]$, $w_i(y_2) = \min(|y_1^* - x_i|, |y_2 - x_i|, |y_3^* - x_i|) \leq \min(|y_1^* - x_i|, |y_3^* - x_i|) = w_i(y_1^*)$. Hence property 1 holds.

This concludes our proof of Claim 1.

Claim 2: There is a solution that optimizes $\max_{\mathbf{y}} \mathrm{SW}(I, \mathbf{y})$ and builds facilities in at most two locations.

Proof: We establish the claim by proving that either $\mathrm{SW}(I, (y_1^*, y_1^*, y_3^*)) \geq \mathrm{SW}(I, \mathbf{y}^*)$ or $\mathrm{SW}(I, (y_1^*, y_3^*, y_3^*)) \geq \mathrm{SW}(I, \mathbf{y}^*)$.

Claim 1 implies that the set of agents $[n]$ can be partitioned into two sets $(S, \overline{S})$ such that $w_i(y_2)$ satisfies property 1 for all i in S, and $w_i(y_2)$ satisfies property 2 for all i in $\overline{S}$. Thus, we have $\mathrm{SW}(y_2) = \sum_{i \in [n]} w_i(y_2) = \sum_{i \in S} w_i(y_2) + \sum_{i \in \overline{S}} w_i(y_2)$. By Lemma 5.5.2, there is a b in $\{y_1^*, y_3^*\}$ such that $\sum_{i \in S} w_i(b) \geq \sum_{i \in S} w_i(y_2)$ for all y_2 in $[y_1^*, y_3^*]$. For any i in $\overline{S}$, we deduce from property 2 that $w_i(b) \geq w_i(y_2)$ for all y_2 in $[y_1^*, y_3^*]$. Therefore, $\mathrm{SW}(b) \geq \mathrm{SW}(y_2)$ for all y_2 in $[y_1^*, y_3^*]$. This completes our proof of Claim 2.

Having established Claim 2, we can assume without loss of generality that $y_2^* = y_3^*$. A similar argument as above can be used to prove that either

$(0, y_2^*, y_2^*)$ or (y_2^*, y_2^*, y_2^*) is an efficient solution. Now if $(0, y_2^*, y_2^*)$ is efficient, then one can use a similar argument to prove that either $(0, 0, 0)$ or $(0, 1, 1)$ is efficient. And if (y_2^*, y_2^*, y_2^*) is efficient, then by applying Lemma 5.5.3 with $k = 3$, we deduce that either $(0, 0, 0)$ or $(1, 1, 1)$ is efficient. Thus, there is a 0-1 efficient solution. The efficiency of Mechanism 2 follows. □

When $k = 2$ (resp., 1), we can add one (resp., two) dummy facilities and use Theorem 5.5.4 to establish that Mechanism 2 is efficient for $k = 2$ (resp., 1). Theorem 5.5.5 below provides a lower bound on the approximation ratio of any SGSP efficient mechanism; this result implies that Mechanism 2 is not SGSP.

Theorem 5.5.5. *There is no SGSP α-efficient DOFLG mechanism with $\alpha < 5/4$.*

Proof. Let n be a large even integer. We construct two $(\frac{3n}{2} + 1)$-agent single-facility DOFLG instances (I, I) and (I, I'). In both (I, I) and (I, I'), agent 1 is located at 0 and dislikes $\{F_1\}$, $n/2$ agents are located at 1 and dislike $\{F_1\}$, and the remaining n agents, which we denote by the set U, are located at 0 and dislike $\emptyset$. In I, all agents report truthfully, while in I', all agents in U report $\{F_1\}$ and the remaining agents report truthfully.

Let the maximum social welfare for instances I and I' be OPT and OPT$'$, respectively. It is easy to see that OPT $= 3n/2$ and OPT$' = n + 1$ (obtained by building F_1 at 0 and 1, respectively). Let the social welfare

achieved by some SGSP DOFLG mechanism M on these instances be ALG and ALG$'$, respectively.

Let M build F_1 at y on I. It follows that $\text{ALG} = y + \frac{3n}{2} - \frac{ny}{2}$. If the agents in U and agent 1 form a coalition in I and the agents in U report $\{F_1\}$, then the instance becomes I'. Thus, as M is SGSP, M cannot build F_1 to the right of y in I'. Using this fact, it is easy to see that $\text{ALG}' \leq (n+1)y + \frac{n}{2}(1-y) = \frac{ny}{2} + \frac{n}{2} + y$.

Using $\text{OPT} = \frac{3n}{2}$ and $\text{ALG} = y + \frac{3n}{2} - \frac{ny}{2}$, we obtain

$$\alpha \geq \frac{\frac{3n}{2}}{y + \frac{3n}{2} - \frac{ny}{2}}. \tag{5.1}$$

Similarly, using $\text{OPT}' = n+1$ and $\text{ALG}' \leq \frac{ny}{2} + \frac{n}{2} + y$, we obtain

$$\alpha \geq \frac{n+1}{\frac{ny}{2} + \frac{n}{2} + y}. \tag{5.2}$$

Let $f(y)$ denote

$$\max\left(\frac{\frac{3n}{2}}{y + \frac{3n}{2} - \frac{ny}{2}}, \frac{n+1}{\frac{ny}{2} + \frac{n}{2} + y}\right).$$

From (5.1) and (5.2) we deduce that $\alpha \geq f(y)$. Let y^* denote a value of y in $[0,1]$ minimizing $f(y)$. It is easy to verify that y^* satisfies $f(y^*) = \frac{5n^2+4n-4}{4n(n+1)}$. Thus, $\alpha \geq f(y^*)$. As n approaches infinity, $f(y^*)$ approaches $5/4$. Thus, for any SGSP α-efficient mechanism, we have $\alpha \geq 5/4$. □ □

In view of Theorem 5.5.5, it is natural to try to determine the minimum value of α for which an SGSP α-efficient DOFLG mechanism exists. Below we present a 2-efficient SGSP mechanism. It remains an interesting open problem

to improve the approximation ratio of 2, or to establish a tighter lower bound for the approximation ratio.

Mechanism 3. Let $(n, k, \mathbf{x}, \mathbf{a})$ denote the reported DOFL instance. Build all facilities at 0 if $\sum_{i\in[n]} x_i \geq \sum_{i\in[n]}(1 - x_i)$; otherwise, build all facilities at 1.

Theorem 5.5.6. *Mechanism 3 is SGSP.*

Proof. Reported dislikes do not affect the locations at which the facilities are built. Hence the theorem follows. □

Theorem 5.5.7. *Mechanism 3 is 2-efficient.*

Proof. Let $I = (n, k, \mathbf{x}, \mathbf{a})$ denote the reported DOFL instance. Let ALG denote the social welfare obtained by Mechanism 3 on this instance, and let OPT denote the maximum possible social welfare on this instance. We need to prove that $2 \cdot \text{ALG} \geq \text{OPT}$.

Assume without loss of generality that Mechanism 3 builds all facilities at 0. (A symmetric argument handles the case where all facilities are built at 1). Then the welfare of an agent i not in indiff(I) is x_i and the welfare of an agent i' in indiff(I) is $\max(x_{i'}, 1 - x_{i'}) \geq x_{i'}$. Thus, $\text{ALG} \geq \sum_{i\in[n]} x_i$. As Mechanism 3 builds the facilities at 0 and not 1, we have $\sum_{i\in[n]} x_i \geq \sum_{i\in[n]}(1 - x_i)$, which implies that $\sum_{i\in[n]} x_i \geq n/2$. Combining the above two inequalities, we have $\text{ALG} \geq n/2$. Since no agent has welfare greater than 1, we have $n \geq \text{OPT}$. Thus, $2 \cdot \text{ALG} \geq n \geq \text{OPT}$, as required. □

We now establish that the analysis of Theorem 5.5.7 is tight by exhibiting a two-facility DOFL instance on which Mechanism 3 achieves half of the optimal social welfare. For the reported DOFL instance $I = (2, 2, (0, 1), (\{F_1\}, \{F_2\}))$, it is easy to verify that the optimal social welfare is $\mathrm{SW}(I, (1, 0)) = 2$, while the social welfare obtained by Mechanism 3 is $\mathrm{SW}(I, (0, 0)) = 1$.

Chapter 6

Egalitarian Resource Sharing Over Multiple Rounds

It is often beneficial for agents to pool their resources in order to better accommodate fluctuations in individual demand. Many multi-round resource allocation mechanisms operate in an online manner: in each round, the agents specify their demands for that round, and the mechanism determines a corresponding allocation. In this chapter, we focus instead on the offline setting in which the agents specify their demand for each round at the outset. We formulate a specific resource allocation problem in this setting, and design and analyze an associated mechanism based on the solution concept of lexicographic maximin fairness. We present an efficient implementation of our mechanism, and prove that it is Pareto-efficient, envy-free, non-wasteful, resource monotonic, population monotonic, and group strategyproof. We also prove that our mechanism guarantees each agent at least half of the utility that they can obtain by not sharing their resources. We complement these positive results by proving that no maximin fair mechanism can improve on the aforementioned factor of one-half.

6.1 Introduction

In this chapter, we consider a group of agents sharing resources over a set of rounds. Every agent owns a specific fraction of the shared resources. Each agent reports their demand for each round. For a given round, an agent accrues utility equal to the number of units allocated to them, as long as the allocation does not exceed their demand; any allocation beyond this threshold does not provide additional utility. Our goal is to design allocation mechanisms based on the demands and ownership shares of the agents, and the supply of the resources. We assume that no monetary exchange occurs between the agents, as is often the case in resource sharing applications (e.g., within a single organization). Such sharing of computational resources arises in applications related to cluster computing, data centers, and supercomputers.

Strategic (coalitions of) agents can misreport their demands in order to achieve higher utility, often at the expense of other agents. Thus, we seek to design strategyproof (SP) or group strategyproof (GSP) mechanisms that incentivize truthful reporting.

An allocation satisfies the sharing incentives (SI) property if it ensures that each agent achieves utility at least as high as they can obtain by not sharing their resources. The mechanism that allocates resources to the agents in proportion to their relative endowments (ownership shares) is GSP and SI. Such a mechanism can be wasteful in the sense that it can allocate resources to an agent in excess of their demand while leaving the demand of another agent unmet. Thus, we seek to design mechanisms that only produce non-

wasteful (NW) allocations. We also seek mechanisms that always produce a Pareto-efficient (PE) allocation.

Given an allocation, we say that agent a envies agent a' if a prefers the allocation of a' (scaled to account for the relative endowments of a and a') to their own. An allocation is envy-free (EF) if no agent envies another. An allocation is frugal if it does not allocate more resources to an agent than they demand. We consider two notions of fair allocations: lexicographically maximin fair (LMMF), and a weaker notion, maximin fair (MMF).

We seek mechanisms that are resource monotonic (RM), that is, if the supply of one or more resources is increased, no agent experiences a decrease in utility. We seek mechanisms that are also population monotonic (PM), that is, if the endowment of one or more agents is decreased, no other agent experiences a decrease in utility.

In this chapter, we present an egalitarian mechanism for allocating resources to agents over multiple rounds and provide an efficient algorithm to compute the allocation. Our mechanism is frugal, LMMF, GSP, 1/2-SI (a relaxation of SI), NW, PE, EF, RM, and PM. We also show that there is no MMF α-SI mechanism for any $\alpha > 1/2$. The significance of our work is discussed in greater detail in Section 6.1.2. But first, we review relevant prior work in Section 6.1.1.

6.1.1 Related Work

Computational Resource Sharing Problem related to computational resource allocation lie at the intersection of economics and computer science, and have received a lot of attention in the research literature. In particular, the theory of fair division, including such concepts as the egalitarian equivalent rule, provides a suitable framework for tackling modern technological challenges arising in cloud computing environments. This connection has inspired a substantial line of theoretical work, including mechanisms for coping with fluctuating demands [59, 83, 160], and for allocating multiple resource types [88, 89] when agents do not know their resource demands [101], and when there is a stream of resources [9].

Some widely used online schedulers (e.g., the fair scheduler implemented in Hadoop and Spark) enforce LMMF. In the online setting, there are two senses in which we can seek to achieve the LMMF property: static and dynamic. In the static sense, we produce an LMMF allocation for each round independently. In the dynamic sense, we produce an allocation for any given round that enforces LMMF over the entire history up to that round (subject to the constraint that the allocations determined for previous rounds cannot be changed).

Our work is inspired by Freeman et al. [83], who studied the game-theoretic aspects of online resource sharing, with a primary focus on the SP, SI, and NW properties. They prove that the static version discussed above satisfies these desiderata, while the dynamic version fails to satisfy SI and SP.

They then consider a more general utility function, where agents derive a fixed "high" utility per unit of resource up to their demands and a fixed "low" utility beyond that threshold. With this utility function, Freeman et al. show that the three aforementioned properties are incompatible in a dynamic setting and thus appropriate trade-offs need to be considered. They propose two mechanisms that partly satisfy the desiderata. Hossain [93] has subsequently presented another mechanism for this setting.

Kandasamy et al. [101] study mechanism design for online resource sharing when agents do not know their resource requirements. Like Freeman et al., they focus on satisfying the SP, SI, and NW properties. Tang et al. [160] propose a dynamic allocation policy for the online setting that is similar to the dynamic version of LMMF discussed above.

Lexicographic Maximin Solutions A lexicographic maximin solution maximizes the minimum utility, and subject to this, maximizes the second-lowest utility, and subject to this, maximizes the third-lowest utility, and so on.

Lexicographic maximin solutions have been studied in many area of research, including computing the nucleolus of cooperative games [146], combinatorial optimization [27], network flows [128, 129], and as one of the standard fairness concepts in telecommunications and network applications [134, 143]. For more details, we refer the reader to the recent work of Ogryczak et al. [133, Section 2.1]. Below we briefly discuss three of these areas that are most rele-

vant to our work.

Multiperiod Resource Allocation Offline resource sharing has been studied in the context of multiperiod resource allocation with equal agent endowments [110, 123]. This line of research is focused on the design of efficient algorithms for computing a lexicographic maximin solution (via linear programming), as opposed to analyzing the associated game-theoretic properties.

Random Assignment Bogomolnaia and Moulin [38] study random assignment problems with dichotomous preferences from a game-theoretical perspective. Dichotomous preferences can be viewed as a special case of fractional demands. Bogomolnaia and Moulin consider several mechanisms, including the LMMF mechanism. They prove that the latter mechanism is GSP, EF, RM, PM, and fair-share (the special case of SI where all demands are zero or infinite). Compared with the model of Bogomolnaia and Moulin, our setting allows for fractional demands, unequal agent endowments, and an unequal supply of resources from one round to the next.

The work of Katta and Sethuraman [103] addresses the random assignment problem with general agent preferences (i.e., where arbitrary indifference are allowed in the agent preferences). They use parametric network flow to achieve LMMF for the special case of dichotomous preferences. For general preferences, they extend the parametric flow algorithm to compute an EF, ordinally efficient assignment, and they prove that no mechanism is SP, EF, and ordinally efficient.

Computer Systems Ghodsi et al. [89] consider the LMMF mechanism in the context of a random assignment problem where the agent endowments need not be the same. Our work strengthens their SP result to GSP while allowing for fractional demands.

Allocation with Substitutable Resources Offline resource sharing can be viewed as the problem of allocating different kinds of substitutable resources to different populations of agents. This allocation problem has been studied in various specific settings, e.g., distribution of coal among power companies [45], multiperiod manufacturing of high-tech products [109], and allocation of vaccines to different populations [155]. To the best of our knowledge, the prior work in this area studies this allocation problem from a computational perspective, rather than a game-theoretic perspective. Sethuraman's survey paper on house allocation problems [132] discusses the connection between allocation with substitutable resources and random assignment with dichotomous preferences.

6.1.2 Significance of Our Work

Freeman et al. [83] study the game-theoretic properties of several online resource allocation mechanisms: the previously known static and dynamic LMMF mechanisms, and the newly-proposed Flexible Lending and T-period mechanisms [83]. In settings where future demands are known, or can be accurately estimated, we can hope to significantly improve upon the fairness guarantees of such online mechanisms. As a simple example, consider an

instance with n agents and n rounds, where each agent contributes a single unit per round. Suppose agents 2 through n each demand two units in every round, and agent 1 demands n units in round 1 and no units thereafter. Clearly, an egalitarian allocation gives a utility of n to every agent. On the other hand, all of the aforementioned online mechanisms give agent 1 a utility of 1, which is only a $1/n$ fraction of the egalitarian share. Our first main contribution is to provide an efficient implementation of a suitable egalitarian mechanism for the offline setting (i.e., where future demands are known); see the first part of Section 6.3.

Our second main contribution is to establish various fundamental game-theoretic properties of our egalitarian mechanism. To do so, we leverage a connection between the random assignment problem of Bogomolnaia and Moulin [38] and our resource sharing problem, Specifically, the special case of their random assignment work in which agents have dichotomous preferences corresponds to the special case of our setting in which the fractional demands of the agents are all 0 or 1. While the work of Bogomolnaia and Moulin provided us with an invaluable roadmap, we found that we still had to overcome some significant technical challenges in order to handle arbitrary fractional demands. (We also handle unequal agent endowments and unequal supplies over the rounds, but generalizing our results in these directions proved to be quite straightforward.) In Section 6.3.1, we establish a number of useful structural properties of lexicographic maximin allocations. In Section 6.3.2, we use these structural properties to establish various game-theoretic properties of

"frugal" LMMF allocations. In Section 6.3.3, we establish that our egalitarian mechanism (and in fact any frugal LMMF mechanism) is GSP.

Our third main contribution is to establish possibility and impossibility results related to the SI property. The SI property is of particular importance in the setting of resource sharing, where we need to ensure that agents are not discouraged from pooling their resources. In Section 6.3.2, we show that any frugal LMMF allocation is $\frac{1}{2}$-SI. In Section 6.4, we show that, for any $\alpha > \frac{1}{2}$, no mechanism is MMF and α-SI. Since no mechanism is MMF and SI, we consider a natural relaxation: mechanisms that are MMF subject to being SI. (In other words, we require the mechanism to be SI, and we only enforce the MMF property with respect to the set of SI allocations.) In Section 6.4, we show that no such mechanism is SP.

6.2 Preliminaries

For any set of agents A, we define endowments(A) as the set of all endowment functions $e : A \to \mathbb{R}_{>0}$, and for any subset A' of A we define $e(A')$ as $\sum_{a \in A'} e(a)$. For any set of objects B, we define supplies(B) as the set of all supply functions $s : B \to \mathbb{R}_{\geq 0}$. Remark: In a multi-round application where z_i units of a resource are allocated in round i, we model the resources associated with round i as an object with supply z_i.

For any set of agents A and any set of objects B, we define demands(A, B)

In the remainder of the chapter, we implicitly define similar overloads for a number of other functions associated with supply, demand, allocation, flow, and capacity.

as the set of all demand functions $d : A \times B \to \mathbb{R}_{\geq 0}$. For any subset A' of A and any d in demands(A, B), $d_{A'}$ denotes the demand function in demands(A', B) such that $d_{A'}(a, b) = d(a, b)$ for all agents a in A' and all objects b in B.

For any set of agents A, any set of objects B, any e in endowments(A), any s in supplies(B), and any d in demands(A, B), the tuple (A, B, e, s, d) denotes an instance of object allocation with fractional demands (OAFD). We think of each agent a in A as owning a $e(a)/e(A)$ fraction of each object in B.

For any OAFD instance $I = (A, B, e, s, d)$, we define allocs(I) as the set of all allocation functions $\mu : A \times B \to \mathbb{R}_{\geq 0}$ such that $\sum_{a \in A} \mu(a, b) \leq s(b)$ for all objects b in B.

An OAFD mechanism M takes as input an OAFD instance I and outputs a subset $M(I)$ of allocs(I).

For any OAFD instance $I = (A, B, e, s, d)$, and any μ in allocs(I), we define the utility of agent a from object b as $u(\mu, d, a, b) = \min(\mu(a, b), d(a, b))$. We assume that the utility of any agent a, denoted $u(\mu, d, a)$, is equal to $\sum_{b \in B} u(\mu, d, a, b)$.

For any OAFD instance $I = (A, B, e, s, d)$, any μ in allocs(I), and any

---

In the present chapter, it is convenient to assume that the output of an OAFD mechanism is a set of allocations, as opposed to a single allocation, because the OAFD mechanism $\mathcal{M}$ that we present in Section 6.3 has this characteristic. For any OAFD instance $I = (A, B, e, s, d)$, all of the agents in A are indifferent between the allocations in $\mathcal{M}(I)$. For any OAFD instance I, our efficient implementation of $\mathcal{M}$ computes a single allocation in $\mathcal{M}(I)$.

b in B, the definition of utility implies that

$$\sum_{a \in A} u(\mu, d, a, b) \leq \min(s(b), \sum_{a \in A} d(a, b)).$$

We let s_I in supplies(I) denote the supply: $s_I(b) = \min(s(b), \sum_{a \in A} d(a, b))$ for all objects b in B. For any OAFD instance $I = (A, B, e, s, d)$, any μ in allocs(I), and any subset A' of A, the definition of utility also implies that $\sum_{a \in A'} u(\mu, d, a) \leq \sum_{b \in B} \min(s_I(b), \sum_{a \in A'} d(a, b))$. We let cap($I, A'$) denote $\sum_{b \in B} \min(s_I(b), \sum_{a \in A'} d(a, b))$.

We now discuss some game-theoretic desiderata for allocations. For any OAFD instance $I = (A, B, e, s, d)$, an allocation μ in allocs(I) is (proportionally) EF if $u(\mu, d, a) \geq \sum_{b \in B} \min(\frac{e(a)}{e(a')} \mu(a', b), d(a, b))$ for all agents a and a' in A. Intuitively, no agent prefers the appropriately scaled (i.e., taking into account relative endowments) version of another agent's allocation to their own allocation.

An allocation μ is said to be PE if there is no allocation μ' such that every agent weakly prefers μ' to μ and some agent prefers μ' to μ. Formally, for any OAFD instance $I = (A, B, e, s, d)$, we say that an allocation μ in allocs(I) is PE if there is no μ' in allocs(I) such that $u(\mu', d, a) \geq u(\mu, d, a)$ for all agents a in A and $u(\mu', d, a) > u(\mu, d, a)$ for some agent a in A.

The SI property requires that any agent a who provides a truthful report achieves utility at least as high as they would achieve with an $e(a)/e(A)$ fraction of every object. Formally, for any OAFD instance $I = (A, B, e, s, d)$ and any α in $[0, 1]$, an allocation μ in allocs(I) is said to be α-SI if $u(\mu, d, a) \geq$

$\alpha \sum_{b \in B} \min(\frac{e(a)}{e(A)} s(b), d(a,b))$ for all agents a in A. We say that an allocation is SI if it is 1-SI.

In our model, the maximum utility that an agent a can achieve from an object b is $d(a,b)$; accordingly, in our setting, there is no reason to allocate more than $d(a,b)$ units of object b to agent a. For any OAFD instance $I = (A, B, e, s, d)$, we say that an allocation μ in allocs(I) is frugal if $\mu(a,b) \leq d(a,b)$ for all (a,b) in $A \times B$. We let frugal(I) denote the set of all frugal allocations in allocs(I). For any OAFD instance $I = (A, B, e, s, d)$ and any μ in frugal(I), we say that μ is NW if for any object b in B, either $\sum_{a \in A} \mu(a,b) = s(b)$ or $\mu(a,b) = d(a,b)$ for all agents a in A.

An allocation μ in allocs(I) is MMF if μ maximizes $\min_{a \in A} u(\mu', d, a)/e(a)$ over all μ' in allocs(I). We let MMF(I) denote the set of all MMF allocations in allocs(I). We let $\mathbf{u}(I, \mu)$ denote the length-$|A|$ vector whose jth component denotes the jth minimum $u(\mu, d, a)/e(a)$ for all agents a in A. An allocation μ in allocs(I) is LMMF if $\mathbf{u}(I, \mu)$ is lexicographically at least $\mathbf{u}(I, \mu')$ for all μ' in allocs(I). We let LMMF(I) denote the set of all LMMF allocations in allocs(I). Note that LMMF is a stricter notion of fairness than MMF.

For any OAFD instance $I = (A, B, e, s, d)$, any subset A' of A, and any μ in LMMF(I), we let sub(I, A', μ) denote the OAFD instance $(A \setminus A', B, e', s', d_{A \setminus A'})$ where $e'(a) = e(a)$ for all agents a in $A \setminus A'$ and $s'(b) = s(b) - \sum_{a \in A'} \mu(a,b)$ for all objects b in B. Lemma 6.2.1 below establishes an optimal substructure property of LMMF allocations.

Lemma 6.2.1. *Let $I = (A, B, e, s, d)$ be an* OAFD *instance, let A' be a subset of A, and let μ belong to* LMMF(I). *Let μ' be the restriction of μ to $A \setminus A'$, that is, $\mu' : (A \setminus A') \times B \to \mathbb{R}_{\geq 0}$ is such that $\mu'(a, b) = \mu(a, b)$ for all (a, b) in $(A \setminus A') \times B$. Then μ' belongs to* LMMF(sub(I, A', μ)).

Proof. For any OAFD instance $\hat{I} = (\hat{A}, \hat{B}, \hat{e}, \hat{s}, \hat{d})$, any $\hat{\mu}$ in LMMF($\hat{I}$), and any subset $\hat{A}'$ of $\hat{A}$, let $\mathbf{u}(\hat{I}, \hat{\mu}, \hat{A}')$ denote the length-$|\hat{A}'|$ vector whose jth component denotes the jth minimum $u(\hat{\mu}, \hat{d}, a)/e(a)$ over all agents a in $\hat{A}'$. Then $\mathbf{u}(\hat{I}, \hat{\mu}) = \text{sort}(\mathbf{u}(\hat{I}, \hat{\mu}, \hat{A}') + \mathbf{u}(\hat{I}, \hat{\mu}, \hat{A} \setminus \hat{A}'))$, where $+$ denotes concatenation and sort is a function that sorts the input vector.

Let $\tilde{I}$ denote sub(I, A', μ). Notice that μ' belongs to allocs($\tilde{I}$). Assume for the sake of contradiction that μ' does not belong to LMMF($\tilde{I}$). Let $\tilde{\mu}$ belong to LMMF($\tilde{I}$). Then $\mathbf{u}(\tilde{I}, \mu') = \mathbf{u}(I, \mu, A \setminus A') \neq \mathbf{u}(\tilde{I}, \tilde{\mu})$. Since $\tilde{\mu}$ belongs to LMMF($\tilde{I}$) and μ' does not belong to LMMF($\tilde{I}$), we deduce that $\mathbf{u}(\tilde{I}, \tilde{\mu})$ is lexicographically greater than $\mathbf{u}(I, \mu, A \setminus A')$.

Let $\mu^* : A \times B \to \mathbb{R}_{\geq 0}$ be defined by $\mu^*(a, b) = \tilde{\mu}(a, b)$ for all (a, b) in $(A \setminus A') \times B$ and $\mu^*(a, b) = \mu(a, b)$ for all (a, b) in $A' \times B$. Thus μ^* belongs to allocs(I) and $\mathbf{u}(I, \mu^*) = \text{sort}(\mathbf{u}(I, \mu^*, A') + \mathbf{u}(I, \mu^*, A \setminus A')) = \text{sort}(\mathbf{u}(I, \mu, A') + \mathbf{u}(\tilde{I}, \tilde{\mu}))$ is lexicographically greater than $\text{sort}(\mathbf{u}(I, \mu, A') + \mathbf{u}(I, \mu, A \setminus A')) = \mathbf{u}(I, \mu)$, a contradiction since μ belongs to LMMF(I). □

For any vectors $\mathbf{v}$ and $\mathbf{w}$ such that $|\mathbf{v}| = |\mathbf{w}| = \ell$, we say that $\mathbf{v}$ Lorenz dominates $\mathbf{w}$ if $\sum_{j=1}^{i} \mathbf{v}_j \geq \sum_{j=1}^{i} \mathbf{w}_j$ for all i in $[\ell]$. If $\mathbf{v}$ Lorenz dominates $\mathbf{w}$, then $\mathbf{v}$ is lexicographically at least $\mathbf{w}$, but the converse does not hold. For

any OAFD instance I and any μ in allocs(I), μ is a Lorenz dominant (LD) allocation if $\mathbf{u}(I, \mu)$ Lorenz dominates $\mathbf{u}(I, \mu')$ for all μ' in allocs(I).

We now discuss some game-theoretic desiderata for OAFD mechanisms. In order to define the SP and GSP properties, it is convenient to first define the k-SP property for any given positive integer k. An OAFD mechanism is k-SP if no coalition of k agents can misrepresent their demands in such a way that some member of the coalition gains and no member of the coalition loses. Formally, an OAFD mechanism M is said to be k-SP if for any OAFD instance $I = (A, B, e, s, d)$, any μ in $M(I)$, any subset A' of A such that $|A'| = k$, any d^* in demands(A', B), any OAFD instance $I' = (A, B, e, s, d')$ where $d' = (d_{A \setminus A'}, d^*)$, and any μ' in $M(I')$, either there is no agent a in A' such that $u(\mu, d, a) < u(\mu', d, a)$, or there is an agent a in A' such that $u(\mu, d, a) > u(\mu', d, a)$. A mechanism is SP if it is 1-SP. A mechanism is GSP if it is k-SP for all k.

An OAFD mechanism is said to be RM if increasing the supply of one or more objects does not decrease the utility of any agent. Formally, an OAFD mechanism M is said to be RM if for any instances $I = (A, B, e, s, d)$, and $I' = (A, B, e, s', d)$ such that $s(b) \leq s'(b)$ for all objects b in B, we have $u(\mu, d, a) \leq u(\mu', d, a)$ for all agents a in A, where μ belongs to $M(I)$ and μ' belongs to $M(I')$.

An OAFD mechanism is said to be PM if decreasing the endowments of one or more agents does not decrease the utility of any other agent. Formally, given any instance $I = (A, B, e, s, d)$, we define shrink(I) as the set of all OAFD

instances $I' = (A', B, e', s, d_{A'})$ such that A' is a subset of A and $e'(a) \leq e(a)$ for all agents a in A'. An OAFD mechanism M is said to be PM if for any OAFD instances $I = (A, B, e, s, d)$ and $I' = (A', B, e', s, d_{A'})$ in shrink(I), any allocations μ in $M(I)$ and μ' in $M(I')$, and any agent a in A' such that $e'(a) = e(a)$, we have $u(\mu, d, a) \leq u(\mu', d, a)$.

An OAFD mechanism M is EF (resp., PE, NW, α-SI, LD) if for any OAFD instance I, every allocation in $M(I)$ is EF (resp., PE, NW, α-SI, LD). An OAFD mechanism M is frugal (resp., MMF, LMMF) if for any OAFD instance I, the set of allocations $M(I)$ is contained in frugal(I) (resp., MMF(I), LMMF(I)).

6.2.1 Lexicographic Flow

In this section, we briefly review the lexicographic flow problem, which we utilize to obtain an efficient implementation of our mechanism.

A flow network is a directed graph $G = (V, E)$ with the vertex set V, the edge set E, having a source vertex s, a sink vertex t, and a non-negative capacity $c(e)$ for each edge e in E. A function $f : E \to R_{\geq 0}$ is said to be a flow if $f(e) \leq c(e)$ (the capacity constraint for edge e) holds for each edge e in E and $\sum_{(u,v)\in E} f(u, v) = \sum_{(v,u)\in E} f(v, u)$ (the flow conservation constraint for vertex v) holds for each vertex v in $V \setminus \{s, t\}$. The value of flow f is defined to be the net flow out of the source s. The goal of the maximum flow problem is to determine a flow of maximum value [79].

A cut of a flow network $G = (V, E)$ is a partition $(S, \overline{S})$ of V such that

s belongs to S and t belongs to $\overline{S}$. The capacity of a cut $(S, \overline{S})$ is defined as the total capacity of all edges going from some vertex in S to some vertex in $\overline{S}$. A minimum cut is a cut of minimum capacity. The famous max-flow min-cut theorem states that in any flow network, the value of a maximum flow is equal to the capacity of a minimum cut. A standard result in network flow theory states that there is a minimum cut $(S, \overline{S})$ such that S contains S' for all minimum cuts $(S', \overline{S'})$. We refer to this minimum cut $(S, \overline{S})$ as the source heavy minimum cut.

In a parametric flow network, each edge capacity is a function of a parameter λ. In this paper, we restrict our attention to parametric flow networks where each edge leaving s has a capacity proportional to λ and all other edge capacities are independent of λ. Parametric flow networks have been widely studied; we refer readers to [85] for more general settings and other results. For any parametric flow network G, we let $G(\lambda)$ denote the flow network associated with a particular value of λ.

In a parametric flow network, the capacity of the minimum cut changes as the value of λ changes. We let the minimum cut capacity function $\kappa(\lambda)$ denote the capacity of the minimum cut as a function of the parameter λ. It is well-known that $\kappa(\lambda)$ is a non-decreasing, concave, and piecewise linear function with at most $|V| - 2$ breakpoints, where a breakpoint is a value of λ at which the slope of $\kappa(\lambda)$ changes [72, 159]. Each of the $|V| - 1$ or fewer line segments that form the graph of $\kappa(\lambda)$ corresponds to a cut. Notice that $\kappa(0) = 0$. As the value of λ increases, the vertices in $V \setminus \{s, t\}$ move from

the sink side to the source side of the source heavy minimum cut. For any parametric flow network G, the breakpoint function $\Lambda(v)$ maps any given vertex v in $V \setminus \{s, t\}$ to the breakpoint value of λ at which v moves from the sink side to the source side of the source-heavy minimum cut. The breakpoint function describes the sequence of cuts associated with $\kappa(\lambda)$ [159].

We now define the notion of a lexicographic flow [128, 129]. Assume that the edges leaving s reach the vertices $\{v_1, \ldots, v_k\}$, and that t does not belong this set. Let the capacity of the edge (s, v_i) be $w_i\lambda$. For a flow f in $G(\infty)$, let $\theta(G, f)$ denote the length-k vector whose jth component is the jth minimum $f(s, v_i)/w_i$, for i in $[k]$. A lexicographic flow f of G is a maximum flow f in $G(\infty)$ that is lexicographically at least $\theta(G, f')$ for all maximum flows f' in $G(\infty)$.

Gallo et al. describe an algorithm that computes the breakpoint function and a lexicographic flow in $O(|V||E| \log(|V|^2/|E|))$ time [85]. This algorithm is slower than the fastest known algorithm for the maximum flow problem [139] by only a logarithmic factor. We describe the algorithm of Gallo et al. here. First, find all the breakpoints of $\kappa(\lambda)$ for a given parametric flow network G. Also, determine the breakpoint $\lambda(v_i)$ for each vertex v_i in $\{v_1, \ldots, v_k\}$ at which v_i moves from the sink side to the source side of the source heavy minimum cut. Let G' be the flow network obtained by setting the capacity of edge (s, v_i) to $w_i\lambda(v_i)$ for each vertex v_i in $\{v_1, \ldots, v_k\}$ in G. Any maximum flow f of G' is a lexicographic flow of G.

6.3 Frugal Lexicographic Maximin Fair Mechanism

Let $\mathcal{M}$ denote the OAFD mechanism such that $\mathcal{M}(I) = \text{LMMF}(I) \cap \text{frugal}(I)$ for all OAFD instances I. In Section 6.3.1 we establish that all agents are indifferent between allocations in $\mathcal{M}(I)$ (see Lemma 6.3.10). In this section, we describe an efficient non-deterministic algorithm $\mathcal{A}$ that implements $\mathcal{M}$ in the following sense: on any input OAFD instance I, the set of possible allocations produced by $\mathcal{A}$ is $\mathcal{M}(I)$. The algorithm $\mathcal{A}$ is based on a reduction to the lexicographic flow problem on a parametric flow network. On input an OAFD instance $I = (A, B, e, s, d)$, algorithm $\mathcal{A}$ first creates a parametric flow network $G_I = (A \cup B \cup \{s, t\}, E)$ with the edge capacities defined by the functions e, d, and s_I described below. The network G_I has an agent (resp., object) vertex for each agent (resp., object) in the input. We denote the set of agent (resp., object) vertices by A (resp., B). For any agent vertex a and any object vertex b, there is a directed edge of capacity $e(a)\lambda$ from s to a, there is a directed edge of capacity $d(a, b)$ from a to b, and there is a directed edge of capacity $s_I(b)$ from b to t. It is easy to check that Observation 6.3.1 below holds.

Observation 6.3.1. *For any* OAFD *instance* $I = (A, B, e, s, d)$*, there is a one-to-one correspondence between flows* f *in* $G_I(\infty)$ *and allocations* μ *in* $\text{frugal}(I)$ *such that* $f(a, b) = \mu(a, b)$ *for all* (a, b) *in* $A \times B$*.*

Given as input an OAFD instance I, algorithm $\mathcal{A}$ non-deterministically selects a lexicographic flow in $G_I(\infty)$ and outputs the corresponding alloca-

tion in allocs(I). The algorithm of Gallo et al. can be used to compute a lexicographic flow in $O((|A| + |B|)|A||B| \log((|A| + |B|)^2/|A||B|))$ time. Using Observation 6.3.1, it is straightforward to prove that there is a one-to-one correspondence between frugal LMMF allocations in $\mathcal{M}(I)$ and lexicographic flows in G_I. Thus we obtain Lemma 6.3.2 below.

Lemma 6.3.2. *For any* OAFD *instance I, the set of possible allocations produced by algorithm $\mathcal{A}$ on input I is equal to $\mathcal{M}(I)$.*

Proof. It is straightforward to verify that the following observations hold.

Observation 6.3.3. *For any* OAFD *instance I, all allocations in* LMMF(I) *are NW.*

Observation 6.3.4. *For any* OAFD *instance I, the capacity of a minimum cut of $G_I(\infty)$ is $s_I(B)$.*

We begin by proving the following useful claim.

Claim 1: Let $I = (A, B, e, s, d)$ be an OAFD instance. Let μ be an allocation in frugal(I) and let f be a flow in $G_I(\infty)$ such that $f(a, b) = \mu(a, b)$ for all (a, b) in $A \times B$. Then f is a maximum flow in $G_I(\infty)$ if and only if μ is NW.

Proof: We first prove the only if direction. Using the max-flow min-cut theorem and Observation 6.3.4, we deduce that the value of flow f is $s_I(B)$. Since μ is frugal and $\mu(a, b) = f(a, b)$ for all (a, b) in $A \times B$, we have $\mu(A, B) = s_I(B)$. Since μ is frugal and $\mu(A, B) = s_I(B)$, we deduce that B,

$\mu(A, b) = s_I(b)$ for all objects b in B, which further implies that μ is NW. Now, we prove the backward direction. Since μ belongs to frugal(I) and μ is NW, we deduce that $\mu(A, B) = s_I(B)$. Thus, the value of flow f is $s_I(B)$. Hence the max-flow min-cut theorem and Observation 6.3.4 imply that flow f is a maximum flow in $G_I(\infty)$. This concludes the proof of Claim 1.

Let $I = (A, B, e, s, d)$ be an OAFD instance. Let $\mathcal{A}(I)$ denote the set of possible allocations produced by algorithm $\mathcal{A}$ on input I. Since $\mathcal{M}(I) = \text{frugal}(I) \cap \text{LMMF}(I)$, it suffices to prove that $\mathcal{A}(I) = \text{frugal}(I) \cap \text{LMMF}(I)$. Thus Claims 2 and 3 below imply that the lemma holds.

Claim 2: $\text{frugal}(I) \cap \text{LMMF}(I) \subseteq \mathcal{A}(I)$.

Proof: Let μ be an allocation in $\text{frugal}(I) \cap \text{LMMF}(I)$. Observation 6.3.1 implies that there is a flow in $G_I(\infty)$, call it f, such that $\mu(a, b) = f(a, b)$ for all (a, b) in $A \times B$. Observation 6.3.3 implies that μ is NW, and hence Claim 1 implies that flow f is a maximum flow in $G_I(\infty)$. To prove that μ belongs to $\mathcal{A}(I)$, it suffices to prove that the f is a lexicographic flow in G_I. Assume for the sake of contradiction that f is not a lexicographic flow in G_I. Hence there is a maximum flow f' in $G_I(\infty)$ such that $\theta(G_I, f')$ is lexicographically greater than $\theta(G_I, f)$. Observation 6.3.1 implies that there is a frugal allocation, call it μ', such that $\mu'(a, b) = f(a, b)$ for all (a, b) in $A \times B$. Since $\mu(a, b) = f(a, b)$ and $\mu'(a, b) = f'(a, b)$ for all (a, b) in $A \times B$, we deduce that $\mathbf{u}(I, \mu) = \theta(G_I, f)$ and $\mathbf{u}(I, \mu') = \theta(G_I, f')$, respectively. Since $\mathbf{u}(I, \mu) = \theta(G_I, f)$, $\mathbf{u}(I, \mu') = \theta(G_I, f')$, and $\theta(G_I, f')$ is lexicographically greater than $\theta(G_I, f)$, we deduce that $\mathbf{u}(I, \mu')$ is lexicographically greater than

$\mathbf{u}(I, \mu)$, a contradiction since μ belongs to LMMF(I). This concludes the proof of Claim 2.

Claim 3: $\mathcal{A}(I) \subseteq \text{frugal}(I) \cap \text{LMMF}(I)$.

Proof: Let μ be an allocation function in $\mathcal{A}(I)$. Let f denote the lexicographic flow in G_I selected by algorithm $\mathcal{A}$; thus f corresponds to μ. Since any lexicographic flow in G_I is a maximum flow in $G_I(\infty)$, we deduce from Observation 6.3.1 that μ belongs to frugal(I). It remains to prove that μ belongs to LMMF(I). Assume for the sake of contradiction that μ does not belong to LMMF(I). Hence there is an allocation μ' in LMMF(I) such that $\mathbf{u}(I, \mu')$ is lexicographically greater than $\mathbf{u}(I, \mu)$. Let μ'' be an allocation in allocs(I) such that $\mu''(a, b) = \min(\mu'(a, b), d(a, b))$ for all (a, b) in $A \times B$. The definition of μ'' implies that μ'' belongs to frugal(I). Since the maximum utility an agent a can achieve from an object b is $d(a, b)$, we have $\mathbf{u}(I, \mu'') = \mathbf{u}(I, \mu')$. Since $\mathbf{u}(I, \mu'') = \mathbf{u}(I, \mu')$ and μ' belongs to LMMF(I), we deduce that μ'' belongs to LMMF(I). Since μ'' belongs to LMMF(I), Observation 6.3.3 implies that μ'' is NW. Since $\mathbf{u}(I, \mu')$ is lexicographically greater than $\mathbf{u}(I, \mu)$ and $\mathbf{u}(I, \mu'') = \mathbf{u}(I, \mu')$, we conclude that $\mathbf{u}(I, \mu'')$ is lexicographically greater than $\mathbf{u}(I, \mu)$. Since μ'' belongs to frugal(I) and μ'' is NW, Observation 6.3.1 and Claim 1 imply that there is a maximum flow in $G_I(\infty)$, call it f'', such that $f''(a, b) = \mu''(a, b)$ for all (a, b) in $A \times B$. Since $\mu(a, b) = f(a, b)$ and $\mu''(a, b) = f''(a, b)$ for all (a, b) in $A \times B$, we deduce that $\mathbf{u}(I, \mu) = \theta(G_I, f)$ and $\mathbf{u}(I, \mu'') = \theta(G_I, f'')$, respectively. Since $\mathbf{u}(I, \mu) = \theta(G_I, f)$, $\mathbf{u}(I, \mu'') = \theta(G_I, f'')$, and $\mathbf{u}(I, \mu'')$ is lexicographically greater than $\mathbf{u}(I, \mu)$, we deduce

that $\theta(G_I, f'')$ is lexicographically greater than $\theta(G_I, f)$, a contradiction since f is a lexicographic flow in G_I. This concludes the proof of Claim 3. $\square$

We introduce some notations that are helpful in analysis of mechanism $\mathcal{M}$. The algorithm of Gallo et al. to find the lexicographic flow computes the breakpoint function Λ_I such that $\Lambda_I(a)$ denotes the breakpoint at which agent vertex a moves from the sink side to the source side of the source-heavy minimum cut, for all agent vertices a in A. Let $\text{num}(I)$ denote $|\{\Lambda_I(a) \mid a \in A\}|$. For any i in $[\text{num}(I)]$, let $\text{brkpts}(I, i)$ denote the ith minimum value in $\{\Lambda_I(a) \mid a \in A\}$, and let $\text{agents}(I, i)$ denote the set $\{a \in A \mid \Lambda_I(a) \leq \text{brkpts}(I, i)\}$. We set $\text{agents}(I, 0) = \emptyset$. For any i in $[\text{num}(I)]$, and object b in B, let $\text{cap}(I, i, b)$ denote $s_I(b) - d(\text{agents}(I, i-1), b)$. With $\text{objects}(I, 0)$ defined as $\emptyset$, for any i in $[\text{num}(I)]$, let $\text{objects}(I, i)$ be recursively defined as the union of $\text{objects}(I, i-1)$ and

$$\{b \in B \setminus \text{objects}(I, i-1) \mid d(\text{agents}(I, i) \setminus \text{agents}(I, i-1), b) > \text{cap}(I, i, b)\}.$$

6.3.1 Technical Properties of Mechanism $\mathcal{M}$

In this section, we establish some basic technical results concerning mechanism $\mathcal{M}$. These results are used in Section 6.3.2 (resp., Section 6.3.3) to derive certain game-theoretic properties of frugal LMMF allocations (resp., mechanisms). Throughout this section, let $I = (A, B, e, s, d)$ denote an OAFD

instance and let G denote G_I. We let Λ and k denote the breakpoint function Λ_I and the value $\text{num}(I)$, respectively. For any i in $[k]$, we let λ_i, A_i, and B_i denote $\text{brkpts}(I,i)$, $\text{agents}(I,i)$, and $\text{objects}(I,i)$, respectively. For any i in $[k]$ and any object b in B, we let $c_i(b)$ denote $\text{cap}(I,i,b)$. For any i in $[k]$, and any non-empty subset A' of $A \setminus A_{i-1}$, let $C_i(A')$ denote $\sum_{b \in B \setminus B_{i-1}} \min(c_i(b), d(A',b))$. Notice that for any non-empty subset A' of A, $C_1(A') = \text{cap}(I,A')$.

Consider a sequence of parametric flow networks $G_1, \ldots, G_k$, where G_i is the subgraph of G induced by $(A \setminus A_{i-1}) \cup (B \setminus B_{i-1}) \cup \{s,t\}$, except that for any object vertex b in G_i, the capacity of edge (b,t) is defined to be $c_i(b)$. Remark: It follows easily from Lemma 6.3.6 below that $c_i(b) \geq 0$.

Before proving Lemma 6.3.6 we show Lemma 6.3.5 below which establishes important properties of the minimum breakpoint of any agent vertex in each parametric flow network G_i. Throughout this section, we make the following definitions for all i in $[k]$: Λ_i^* denotes the breakpoint function of G_i; λ_i^* denotes the minimum breakpoint of an agent vertex in G_i; λ_i^{**} denotes $\min_{\emptyset \neq A' \subseteq A \setminus A_{i-1}} C_i(A')/e(A')$; A_i^* denotes $\bigcup\{A' \subseteq A \setminus A_{i-1} \mid C_i(A') = e(A')\lambda_i^*\}$; B_i^* denotes $\{b \in B \setminus B_{i-1} \mid d(A_i^*, b) > c_i(b)\}$; $\Psi_1(i)$ denotes the predicate "$\lambda_i^* = \lambda_i^{**}$"; $\Psi_2(i)$ denotes the predicate "$\Lambda_i^*(a) = \lambda_i^*$ for all agents a in A_i^*"; $\Psi_3(i)$ denotes the predicate "for any flow in $G_i(\infty)$ such that $f(s,a) = e(a)\lambda_i^*$ for all agent vertices a in $A \setminus A_{i-1}$, we have $f(a,b) = d(a,b)$ for all (a,b) in $A_i^* \times ((B \setminus B_{i-1}) \setminus B_i^*)$"; $\Psi_4(i)$ denotes the predicate "for any flow in $G_i(\infty)$ such that $f(s,a) = e(a)\lambda_i^*$ for all agent vertices a in $A \setminus A_{i-1}$, we

have $f(A_i^*, b) = f(b, t) = c_i(b)$ for all object vertices b in B_i^*."

Lemma 6.3.5. *Let i be in $[k]$. Then predicate $\Psi_j(i)$ holds for all j in $\{1, \ldots, 4\}$.*

Proof. Recall that the set of agent (resp., object) vertices in G_i is $A \setminus A_{i-1}$ (resp., $B \setminus B_{i-1}$). We first establish the following useful claim, which implies that $\lambda_i^* \geq \lambda_i^{**}$.

Claim 1: A maximum flow in $G_i(\lambda_i^{**})$ has value $e(A \setminus A_{i-1})\lambda_i^{**}$.

Proof: To prove that a maximum flow in $G_i(\lambda_i^{**})$ has value $e(A \setminus A_{i-1})\lambda_i^{**}$, it is sufficient to argue that a minimum cut in $G_i(\lambda_i^{**})$ has capacity $e(A \setminus A_{i-1})\lambda_i^{**}$. Let the source-heavy minimum cut in $G_i(\lambda_i^{**})$ be $(S, \overline{S})$ and let A' denote $S \cap (A \setminus A_{i-1})$. We begin by showing that $S \cap (B \setminus B_{i-1}) = \{b \in B \setminus B_{i-1} \mid d(A', b) \geq c_i(b)\}$. Assume for the sake of contradiction that this equation does not hold. We consider three cases.

Case 1: There is an object vertex b in $\overline{S} \cap (B \setminus B_{i-1})$ such that $d(A', b) > c_i(b)$. Hence the capacity of the cut $(S + b, \overline{S} - b)$ is $d(A', b) - c_i(b) > 0$ less than the capacity of the cut $(S, \overline{S})$, a contradiction since $(S, \overline{S})$ is a minimum capacity cut.

Case 2: There is an object vertex b in $S \cap (B \setminus B_{i-1})$ such that $c_i(b) > d(A', b)$. Hence the capacity of the cut $(S - b, \overline{S} + b)$ is $c_i(b) - d(A', b) > 0$ less than the capacity of the cut $(S, \overline{S})$, a contradiction since $(S, \overline{S})$ is a minimum capacity cut.

Case 3: There is an object b in $\overline{S} \cap (B \setminus B_{i-1})$ such that $c_i(b) = d(A', b)$.

Hence the cuts $(S+b, \overline{S}-b)$ and $(S, \overline{S})$ have the same capacity, but $(S+b, \overline{S}-b)$ has a larger source side, a contradiction.

From the above case analysis, $S \cap (B \setminus B_{i-1}) = \{b \in B \setminus B_{i-1} \mid d(A', b) \geq c_i(b)\}$. Thus the capacity of the cut $(S, \overline{S})$ is

$$\sum_{a \in (A \setminus A_{i-1}) \setminus A'} e(a)\lambda_i^{**} + \sum_{b \in B \setminus B_{i-1}} \min(c_i(b), d(A', b)) = e((A \setminus A_{i-1}) \setminus A')\lambda_i^{**} + C_i(A').$$

The definition of λ_i^{**} implies that $e(A')\lambda_i^{**} \leq C_i(A')$. Thus the capacity of the minimum cut is at least $e(A \setminus A_{i-1})\lambda_i^{**}$. Moreover, the capacity of cut $(s, V \setminus s)$ is $e(A \setminus A_{i-1})\lambda_i^{**}$. Thus the capacity of a minimum cut of $G_i(\lambda_i^{**})$ is $e(A \setminus A_{i-1})\lambda_i^{**}$. This concludes the proof of Claim 1.

Let λ' be a value greater than λ_i^{**}. We show that there is no flow in $G_i(\lambda')$ such that every agent vertex a has incoming flow $e(a)\lambda'$. Assume for the sake of contradiction that there is a flow such that every agent vertex a has incoming flow $e(a)\lambda'$. The total capacity of the edges leaving $A_i^* \cup B_i^*$ is

$$\sum_{b \in B \setminus B_{i-1}} \min(c_i(b), d(A_i^*, b)) = C_i(A_i^*).$$

Since $C_i(A_i^*) = e(A_i^*)\lambda_i^{**} < e(A_i^*)\lambda'$, the total capacity of the edges leaving $A_i^* \cup B_i^*$ is less than the total flow into the set $A_i^* \cup B_i^*$, a contradiction. This result, together with Claim 1, establishes that $\Psi_1(i)$ and $\Psi_2(i)$ hold.

Let f be a flow in $G_i(\infty)$ such that $f(s, a) = e(a)\lambda_i^*$ for all agent vertices a in $A \setminus A_{i-1}$ Since the total capacity of the edges leaving $A_i^* \cup B_i^*$ is $C_i(A_i^*)$, which is equal to the total flow $e(A_i^*)\lambda_i^*$ into $A_i^* \cup B_i^*$ in f, we deduce that $f(e) = c(e)$ for all edges e leaving $A_i^* \cup B_i^*$. Thus $f(b, t) = c_i(b)$ for all object vertices

b in B_i^*, and $f(a,b) = d(a,b)$ for all (a,b) in $A_i^* \times ((B \setminus B_{i-1}) \setminus B_i^*)$. Moreover, since the total flow into $A_i^* \cup B_i^*$ is $C_i(A_i^*) = c_i(B_i^*) + d(A_i^*, (B \setminus B_{i-1}) \setminus B_i^*)$, we have $f(A_i^*, B_i^*) = c_i(B_i^*)$. It follows that $f(A_i^*, b) = f(b,t)$ for all object vertices b in B_i^*. We conclude that $\Psi_3(i)$ and $\Psi_4(i)$ hold. □

Lemma 6.3.6 below characterizes the values of the breakpoints of G, and the breakpoint associated with each agent vertex. It also establishes a connection between a lexicographic flow in G and the sets $A_1, \ldots, A_k, B_1, \ldots, B_k$. The result of Lemma 6.3.6 is similar in spirit to Theorem 4.6 of Megiddo [128] for general parametric flow networks. Since we work with parametric flow networks with a special structure, we are able to obtain a more specific result and we can characterize a lexicographic flow in greater detail. Our proof of Lemma 6.3.6 is not based on Megiddo's proof; instead, we provide a simpler proof for our special case. Our formulation of Lemma 6.3.6 generalizes Megiddo's result in one aspect, since it allows for agents with different endowments; this generalization is straightforward.

For any i in $[k]$, we define the following predicates: $\Gamma_1(i)$ denotes "the minimum breakpoint of any agent vertex in G_i is λ_i"; $\Gamma_2(i)$ denotes "λ_i is equal to $\min_{\emptyset \neq \tilde{A} \subseteq A \setminus A_{i-1}} C_i(\tilde{A})/e(\tilde{A})$"; $\Gamma_3(i)$ denotes "A_i is equal to $A_{i-1} \cup \bigcup\{\tilde{A} \subseteq A \setminus A_{i-1} \mid C_i(\tilde{A}) = e(\tilde{A})\lambda_i\}$"; $\Gamma_4(i)$ denotes "for any lexicographic flow f in G, we have $f(a,b) = d(a,b)$ for all (a,b) in $(A_i \setminus A_{i-1}) \times (B \setminus B_i)$"; $\Gamma_5(i)$ denotes "for any lexicographic flow f in G, we have $f(A_i, b) = f(b,t) = s_I(b)$ and $f(a,b) = 0$ for all (a,b) in $(A \setminus A_i) \times (B_i \setminus B_{i-1})$."

Lemma 6.3.6. *Predicate $\Gamma_j(i)$ holds for all i in $[k]$ and all j in $\{1,\ldots,5\}$.*

Proof. Let f denote a lexicographic flow in G and for any i in $[k]$, and let $P(i)$ denote the predicate "$\Gamma_j(i)$ holds for all j in $\{1,\ldots,5\}$." We prove by induction that $P(i)$ holds for all i in $[k]$.

Base case: Since $A_0 = \emptyset$, we have $c_1(b) = s_I(b)$ for all object vertices b in B. Thus $G_1 = G$, and hence $\Gamma_1(1)$ holds. Lemma 6.3.5 implies that $\Psi_1(1)$ and $\Psi_2(1)$ hold; hence $\Gamma_2(1)$ and $\Gamma_3(1)$ hold. Since $\Psi_3(1)$ and $\Psi_4(1)$ hold by Lemma 6.3.5, $\lambda_1^* = \lambda_1$, $A_1^* = A_1 \setminus A_0$, and $B_1^* = B_1 \setminus B_0$, we deduce that $\Gamma_4(1)$ and $\Gamma_5(1)$ hold.

Induction step: Let i belong to $\{2,\ldots,k\}$ and assume that $P(i')$ holds for all i' in $[i-1]$. We need to prove that $P(i)$ holds. Let b be an object vertex in $B \setminus B_{i-1}$. Since the IH implies that $\Gamma_4(i')$ holds for all i' in $[i-1]$, we deduce that $f(a,b) = d(a,b)$ for all a in A_{i-1}. Thus $f(A \setminus A_{i-1}, b) \leq s_I(b) - d(A_{i-1}, b) = c_i(b)$. Moreover, since the IH implies that $\Gamma_5(i')$ holds for all i' in $[i-1]$, we deduce that $f(a,b') = 0$ for all (a,b') in $(A \setminus A_{i-1}) \times B_{i-1}$. From the aforementioned results, it is straightforward to verify that $\Gamma_1(i)$ holds. Lemma 6.3.5 implies that $\Psi_1(i)$ and $\Psi_2(i)$ hold; hence $\Gamma_2(i)$ and $\Gamma_3(i)$ hold. Since $\Psi_3(i)$ holds by Lemma 6.3.5, $\lambda_i^* = \lambda_i$, $A_i^* = A_i \setminus A_{i-1}$, and $B_i^* = B_i \setminus B_{i-1}$, we deduce that $\Gamma_4(i)$ holds. Let b' be an object vertex in $B_i \setminus B_{i-1}$. Predicate $\Psi_4(i)$ implies that $f(A_i \setminus A_{i-1}, b') = c_i(b') = s_I(b') - d(A_{i-1}, b')$. Since the IH implies $\Gamma_4(i')$ holds for all i' in $[i-1]$, we deduce that $f(a,b') = d(a,b')$ for all a in A_{i-1}. Hence $f(A_{i-1}, b') = d(A_{i-1}, b')$. Since $f(A_{i-1}, b') = d(A_{i-1}, b')$ and

$f(A_i \setminus A_{i-1}, b') = s_I(b') - d(A_{i-1}, b')$, we deduce that $f(A_i, b') = s_I(b') = f(b', t)$, where the last equality holds because the capacity of edge (b', t) is $s_I(b')$. Since $f(A_i, b') = s_I(b')$ and $s_I(b')$ is the capacity of edge (b', t), we deduce that $f(a, b) = 0$ for all a in $A \setminus A_i$, which establishes $\Gamma_5(i)$. We conclude that $P(i)$ holds, as required. □

Corollary 6.3.7 below is easy to verify.

Corollary 6.3.7. *For any lexicographic flow f in G and any maximum flow f' in $G(\infty)$, $\theta(G, f)$ Lorenz dominates $\theta(G, f')$.*

We now prove some results about frugal LMMF allocations. Throughout the remainder of the section, let μ denote an allocation in $\mathcal{M}(I)$. The definition of mechanism $\mathcal{M}$ implies that μ is frugal and LMMF. Recall that algorithm $\mathcal{A}$ first computes a lexicographic flow f in G such that $\mu(a, b) = f(a, b)$ for all (a, b) in $A \times B$.

Corollaries 6.3.8 and 6.3.9 below describe the structural properties of the allocation μ and immediately follow from predicates Γ_4 and Γ_5, respectively.

Corollary 6.3.8. *For any i in $[k]$, any agent a in $A_i \setminus A_{i-1}$, and any object b in $B \setminus B_i$, we have $\mu(a, b) = d(a, b)$.*

Corollary 6.3.9. *For any i in $[k]$, any agent a in $A \setminus A_i$, and any object b in B_i, we have $\mu(a, b) = 0$.*

Lemma 6.3.10 below establishes some basic results that are useful for many of our subsequent proofs. For example, we use Lemma 6.3.10 along with Corollaries 6.3.8 and 6.3.9 to prove that any frugal LMMF allocation is EF (Theorem 6.3.17).

Lemma 6.3.10. *Let a be an agent in A and let b be an object B. Then, $\mu(a,b)$ belongs to $[0, d(a,b)]$, and $u(\mu, d, a) = \mu(a, B) = e(a)\Lambda(a)$.*

Proof. The capacity of edge (a,b) in G is $d(a,b)$. Hence $\mu(a,b) = f(a,b)$ belongs to $[0, d(a,b)]$. Flow f satisfies $e(a)\Lambda(a) = f(s,a) = f(a,B) = \mu(a,B) = u(\mu,d,a)$. □

We use Lemma 6.3.11 below, along with Lemma 6.3.10 and Corollary 6.3.8, to prove that any frugal LMMF allocation is 1/2-SI (Theorem 6.3.18).

Lemma 6.3.11. *Let i be in $[k]$. Then $\sum_{j\in[i]} e(A_j \setminus A_{j-1})\lambda_j = s(B_i) + d(A_i, B \setminus B_i)$.*

Proof. We begin by proving the following useful claim.

Claim 1: Let i belong to $[k]$. Let b be an object in $B_i \setminus B_{i-1}$. Then $s_I(b) = s(b)$.

To prove Claim 1, observe that the definition of $B_i \setminus B_{i-1}$ implies that $d(A_i \setminus A_{i-1}, b) > c_i(b) = s_I(b) - d(A_{i-1}, b)$. Thus $d(A,b) \geq d(A_i, b) > s_I(b)$. Since $d(A,b) > s_I(b)$ and $s_I(b) = \min(s(b), d(A,b))$, we have $s_I(b) = s(b)$. This completes the proof of Claim 1.

Notice that $\sum_{j\in[i]} e(A_j \setminus A_{j-1})\lambda_j$ is the total flow into A_i in f. The total flow out of A_i in f is $f(A_i, B)$. For any agent vertex a in A_i and any object vertex b in $B \setminus B_i$, Lemma 6.3.6 implies that $f(a, b) = d(a, b)$. For any object b in B_i, Lemma 6.3.6 implies that $f(A_i, b) = s_I(b)$. Thus $f(A_i, B) = s_I(B_i) + d(A_i, B \setminus B_i)$. Since the net flow into A_i is 0, we obtain $\sum_{j\in[i]} e(A_j \setminus A_{j-1})\lambda_j = s_I(B_i) + d(A_i, B \setminus B_i) = s(B_i) + d(A_i, B \setminus B_i)$, where the last equality follows from Claim 1. □

We use Lemma 6.3.12 below, along with Lemma 6.3.10 and the result that any frugal LMMF allocation is NW (Theorem 6.3.14), to prove that any frugal LMMF mechanism is RM (Theorem 6.3.19). We use Lemmas 6.2.1, 6.3.6, and 6.3.10 and the result that any frugal LMMF mechanism is RM (Theorem 6.3.19) to prove that any frugal LMMF mechanism is PM (Theorem 6.3.20).

Lemma 6.3.12. *Let i be in $[k]$, let a (resp., a') be an agent in A_i (resp., $A \setminus A_i$), and let b be an object in B such that $\mu(a', b) > 0$. Then $\mu(a, b) = d(a, b)$.*

Proof. Corollary 6.3.9 and $\mu(a', b) > 0$ imply that b belongs to $B \setminus B_i$. Hence Corollary 6.3.8 implies that $\mu(a, b) = d(a, b)$. □

We use Lemma 6.3.13 below, along with Lemmas 6.2.1, 6.3.6, and 6.3.10 and the result that any frugal LMMF mechanism is RM (Theorem 6.3.19), to prove that any frugal LMMF mechanism is GSP (Theorem 6.3.21).

Lemma 6.3.13. *Let i belong to $[k]$ and let μ' be an allocation in* allocs(I) *such that $u(\mu', d, a) \geq e(a)\Lambda(a)$ for all agents a in A_i. Then $\mu'(A_i, b) \geq \mu(A_i, b)$ for all objects b in B.*

Proof. For any object b in B, we let $u(b)$ (resp., $u'(b)$) denote $\sum_{a \in A_i} u(\mu, d, a, b)$ (resp., $\sum_{a \in A_i} u(\mu', d, a, b)$). We begin by establishing a useful claim.

Claim 1: We have $u'(b) \leq u(b)$ for all b in B.

Proof: Let b be an object in B. We consider two cases.

Case 1: $b \in B_i$. The definition of $s_I(b)$ implies that $u'(b) \leq s_I(b)$. Using Lemma 6.3.6 and the definition of μ, we deduce that $\mu(A_i, b) = f(A_i, b) = s_I(b)$. Lemma 6.3.10 implies that $\mu(a, b)$ belongs to $[0, d(a, b)]$ for all a in A_i. Using Lemma 6.3.10 we conclude that $u(b) = \mu(A_i, b) = s_I(b)$. Thus $u'(b) \leq u(b)$.

Case 2: $b \in B \setminus B_i$. We have $u'(b) \leq d(A_i, b)$. Using Lemma 6.3.6 and the definition of μ, we deduce that for any agent a in A_i, $\mu(a, b) = f(a, b) = d(a, b)$. Again, using Lemma 6.3.10, we conclude that $u(b) = d(A_i, b)$. Thus, $u'(b) \leq u(b)$.

We have

$$\sum_{b \in B} u'(b) = \sum_{a \in A_i} u(\mu', d, a) \geq \sum_{a \in A_i} \Lambda(a)e(a) = \sum_{a \in A_i} u(\mu, d, a) = \sum_{b \in B} u(b),$$

where the second equality follows from Lemma 6.3.10. Together with Claim 1, we deduce that $u'(b) = u(b)$ for all b in B. Thus $\mu'(A_i, b) \geq u'(b) = u(b) = \mu(A_i, b)$ for all b in B, where the last equality follows from Lemma 6.3.10. □

6.3.2 Game-Theoretic Properties of Frugal LMMF Allocations

In this section we establish some game-theoretic properties of frugal LMMF allocations. Throughout this section let $I = (A, B, e, s, d)$ be an OAFD instance, and let μ belong to $\mathcal{M}(I)$. The definition of $\mathcal{M}$ implies that μ is an arbitrary frugal LMMF allocation. Theorems 6.3.14 through 6.3.18 and Corollaries 6.3.15 and 6.3.16 imply that any frugal LMMF mechanism is NW, PE, LD, EF and 1/2-SI.

Theorem 6.3.14. *Allocation μ is NW.*

Proof. The definition of μ implies that μ belongs to frugal(I). Assume for the sake of contradiction that μ is not NW. Hence there is an agent a in A and an object b in B such that $\mu(a, b) < d(a, b)$ and $\mu(A, b) < s(b)$. Let μ' be the allocation in allocs(I) such that $\mu'(a, b) = \min(d(a, b), s(b) - \mu(A - a, b)) > \mu(a, b)$, and $\mu'(a', b') = \mu(a', b')$ for all (a', b') in $A \times B - (a, b)$. Thus $u(\mu', d, a) > u(\mu, d, a)$ and $u(\mu', d, a') = u(\mu, d, a')$ for all agents a' in $A - a$, a contradiction since μ is LMMF. □

Any NW allocation is also PE, which gives Corollary 6.3.15 below.

Corollary 6.3.15. *Allocation μ is PE.*

Since mechanism $\mathcal{M}$ obtains the allocation from the lexicographic flow in the corresponding parametric graph and Corollary 6.3.7, we obtain Corollary 6.3.16.

Corollary 6.3.16. *Allocation μ is LD.*

Theorem 6.3.17 below shows that any frugal LMMF allocation is EF. Bogomolnaia and Moulin [38] show that any frugal LMMF allocation is EF when all demands are 0 or 1, all agent endowments are equal, and all object supplies are equal. To generalize this result to our setting, the main issue is to handle fractional demands; Corollaries 6.3.8 and 6.3.9 play a primary role in this regard.

Theorem 6.3.17. *Allocation μ is EF.*

Proof. Assume for the sake of contradiction that there are agents a and a' such that agent a envies the allocation of agent a', that is,

$$u(\mu, d, a) < \sum_{b \in B} \min(\frac{e(a)}{e(a')}\mu(a', b), d(a, b)). \tag{6.1}$$

As in Section 6.3.1, let k denote the value $\text{num}(I)$. For any i in $[k]$, let λ_i, A_i, and B_i denote $\text{brkpts}(I, i)$, $\text{agents}(I, i)$, and $\text{objects}(I, i)$, respectively. Let i and i' in $[k]$ be such that agent a (resp. a') belongs to $A_i \setminus A_{i-1}$ (resp., $A_{i'} \setminus A_{i'-1}$). We consider two cases.

Case 1: In this case we have $i' > i$. We deduce that

$$\begin{aligned} u(\mu, d, a) &= \sum_{b \in B} \min(\mu(a,b), d(a,b)) \\ &= \sum_{b \in B \setminus B_i} \min(\mu(a,b), d(a,b)) + \sum_{b \in B_i} \min(\mu(a,b), d(a,b)) \\ &= \sum_{b \in B \setminus B_i} d(a,b) + \sum_{b \in B_i} \min(\mu(a,b), d(a,b)) \\ &\geq \sum_{b \in B \setminus B_i} d(a,b), \end{aligned} \tag{6.2}$$

where the last equality follows from Corollary 6.3.8. Corollary 6.3.9 implies that $\mu(a', b) = 0$ for all objects b in B_i. Therefore,

$$\begin{aligned} \sum_{b \in B} \min(\frac{e(a)}{e(a')}\mu(a', b), d(a,b)) &= \sum_{b \in B \setminus B_i} \min(\frac{e(a)}{e(a')}\mu(a', b), d(a,b)) \\ &\leq \sum_{b \in B \setminus B_i} d(a,b). \end{aligned} \tag{6.3}$$

Inequalities (6.2) and (6.3) imply that

$$\sum_{b \in B} \min(\frac{e(a)}{e(a')}\mu(a', b), d(a,b)) \leq u(\mu, d, a),$$

contradicting inequality (6.1).

Case 2: $i' \leq i$. Since $\lambda_1, \ldots, \lambda_k$ is an increasing sequence, $\lambda_{i'} \leq \lambda_i$. Lemma 6.3.10 implies that $\mu(a', B) = e(a')\lambda_{i'}$. Thus

$$\begin{aligned} \sum_{b \in B} \min(\frac{e(a)}{e(a')}\mu(a', b), d(a,b)) &\leq \frac{e(a)}{e(a')} \sum_{b \in B} \mu(a', b) \\ &= e(a)\lambda_{i'} \\ &\leq e(a)\lambda_i \\ &= u(\mu, d, a), \end{aligned}$$

where the first and second equalities follow from Lemma 6.3.10. This contradicts inequality (6.1). □

Theorem 6.3.18 below shows that any frugal LMMF allocation is 1/2-SI. Lemma 6.4.1 in Section 6.4 implies that no frugal LMMF mechanism is α-SI for any $\alpha > 1/2$.

Theorem 6.3.18. *Allocation μ is 1/2-SI.*

Proof. Let a be an agent in A and let $\mathrm{SI}(a)$ denote $\sum_{b \in B} \min\left(\frac{e(a)}{e(A)} s(b), d(a,b)\right)$. We need to show that $u(\mu, d, a) \geq \mathrm{SI}(a)/2$.

As in Section 6.3.1, let Λ and k denote the breakpoint function Λ_I and the value $\mathrm{num}(I)$, respectively. For any i in $[k]$, let λ_i, A_i, and B_i denote $\mathrm{brkpts}(I, i)$, $\mathrm{agents}(I, i)$, and $\mathrm{objects}(I, i)$, respectively. Let i in $[k]$ be such that agent a belong to $A_i \setminus A_{i-1}$. Thus $\Lambda(a) = \lambda_i$.

Lemma 6.3.10 implies that $u(\mu, d, a) = e(a)\lambda_i$. We have

$$\mathrm{SI}(a) = \sum_{b \in B} \min\left(\frac{e(a)}{e(A)} s(b), d(a,b)\right) \leq \frac{e(a)}{e(A)} s(B_i) + d(a, B \setminus B_i).$$

Thus, to prove that $u(\mu, d, a) \geq \mathrm{SI}(a)/2$, it suffices to prove that $u(\mu, d, a) \geq d(a, B \setminus B_i)$ and $u(\mu, d, a) \geq \frac{e(a)}{e(A)} s(B_i)$. Observe that $u(\mu, d, a) = \mu(a, B) \geq \mu(a, B \setminus B_i) = d(a, B \setminus B_i)$, where the first equality follows from Lemma 6.3.10 and the last equality follows from Corollary 6.3.8. Thus $u(\mu, d, a) \geq d(a, B \setminus B_i)$.

It remains to prove that

$$u(\mu, d, a) \geq \frac{e(a)}{e(A)} s(B_i).$$

dian Since $u(\mu, d, a) = e(a)\lambda_i$, it suffices to prove that $\lambda_i \geq s(B_i)/e(A)$. Note that $e(A_i)\lambda_i \geq \sum_{j \in [i]} e(A_j \setminus A_{j-1})\lambda_j = s(B_i) + d(A_i, B \setminus B_i) \geq s(B_i)$, where the first inequality follows since $\lambda_1, \ldots, \lambda_k$ is an increasing sequence and the first equality follows from Lemma 6.3.11. Since $e(A_i)\lambda_i \geq s(B_i)$, we find that $\lambda_i \geq s(B_i)/e(A_i) \geq s(B_i)/e(A)$, where the last inequality holds because A_i is a subset of A and hence $e(A_i) \leq e(A)$. □

6.3.3 Game-Theoretic Properties of Frugal LMMF Mechanisms

In this section, we establish game-theoretic properties of frugal LMMF mechanisms.

Theorem 6.3.19. *Any frugal LMMF mechanism is RM.*

Proof. The definition of mechanism $\mathcal{M}$ implies that it is sufficient to show that $\mathcal{M}$ is RM. Let $I = (A, B, e, s, d)$ and $I' = (A, B, e, s', d)$ denote OAFD instances such that $s(b) \leq s'(b)$ for all objects b in B, let μ belong to $\mathcal{M}(I)$, and let μ' belong to $\mathcal{M}(I')$. We have to prove that $u(\mu, d, a) \leq u(\mu', d, a)$ for all agents a in A. Let Λ and Λ' denote the breakpoint functions for Λ_I and $\Lambda_{I'}$, respectively. We begin by proving the following useful claim.

Claim 1: Let a and a' be agents in A and let b be an object in B such that $\mu'(a, b) < \mu(a, b)$ and $\mu'(a', b) > \mu(a', b)$. Then $\Lambda(a) \leq \Lambda(a')$ and $\Lambda'(a') \leq \Lambda'(a)$.

To prove Claim 1, first observe that $0 \leq \mu'(a,b) < \mu(a,b) \leq d(a,b)$ and $0 \leq \mu(a',b) < \mu'(a',b) \leq d(a',b)$ by Lemma 6.3.10. Since $\mu(a,b) > 0$ and $\mu(a',b) < d(a',b)$, Lemma 6.3.12 implies that $\Lambda(a) \leq \Lambda(a')$. Similarly, since $\mu'(a',b) > 0$ and $\mu'(a,b) < d(a,b)$, Lemma 6.3.12 implies that $\Lambda'(a') \leq \Lambda'(a)$. This completes the proof of Claim 1.

Let A' denote $\{a \in A \mid u(\mu,d,a) > u(\mu',d,a)\}$. To establish the lemma, we need to prove that A' is empty. Assume for the sake of contradiction that A' is nonempty. Let λ^* denote $\min_{a \in A'} \Lambda'(a)$, and let A'' denote $\{a \in A' \mid \Lambda'(a) = \lambda^*\}$; thus A'' is nonempty.

Let B' denote $\{b \in B \mid \mu(A'',b) > \mu'(A'',b)\}$. The set B' is nonempty since A'' is a nonempty subset of A'. Let b denote an object in B'.

Let A''' denote $\{a \in A'' \mid \mu(a,b) > \mu'(a,b)\}$. The set A''' is nonempty since $\mu(A'',b) > \mu'(A'',b)$. Let a denote an agent in A'''. Since $u(\mu,d,a) = e(a)\Lambda(a)$ and $u(\mu',d,a) = e(a)\Lambda'(a)$ by Lemma 6.3.10, and since a belongs to A', we deduce that $\Lambda(a) > \Lambda'(a) = \lambda^*$.

Since $\mu(A'',b) > \mu'(A'',b)$ and Theorem 6.3.14 implies that $\mu(A,b) = \mu'(A,b)$, we deduce that there is an agent in $A \setminus A''$, call it a', such that $\mu(a',b) < \mu'(a',b)$.

Since $\mu(a,b) > \mu'(a,b)$ and $\mu(a',b) < \mu'(a',b)$, Claim 1 implies that $\Lambda(a) \leq \Lambda(a')$ and $\Lambda'(a') \leq \Lambda'(a) = \lambda^*$. Since $\Lambda(a) > \lambda^*$, we have $\Lambda'(a') \leq \lambda^* < \Lambda(a) \leq \Lambda(a')$. Thus a' belongs to A', and hence the definition of λ^* implies $\Lambda'(a') \geq \lambda^*$. Since $\Lambda'(a') \leq \lambda^*$, we conclude that $\Lambda'(a') = \lambda^*$. Since a' belongs

to A' and $\Lambda'(a') = \lambda^*$, we deduce that a' belongs to A'', a contradiction. $\square$

Theorem 6.3.20. *Any frugal LMMF mechanism is PM.*

Proof. By the definition of mechanism $\mathcal{M}$, it is sufficient to show that $\mathcal{M}$ is PM.

Let $P(k)$ denote the predicate "for any OAFD instances $I = (A, B, e, s, d)$ and $I' = (A', B, e', s, d_{A'})$ such that $|A| = k$ and I' belongs to shrink(I), any allocations μ in $\mathcal{M}(I)$ and μ' in $\mathcal{M}(I')$, and any agent a in A' such that $e'(a) = e(a)$, we have $u(\mu, d, a) \leq u(\mu', d, a)$." We prove by induction that $P(k)$ holds for all $k \geq 0$, which implies that the theorem holds.

Base case: It is easy to see that $P(0)$ holds.

Induction step: Let k be a positive integer and assume that $P(i)$ holds for $0 \leq i < k$. We need to prove that $P(k)$ holds. Let $I = (A, B, e, s, d)$ and $I' = (A', B, e', s, d_{A'})$ be OAFD instances such that $|A| = k$ and I' belongs to shrink(I). Let allocation μ (resp., μ') belong to $\mathcal{M}(I)$ (resp., $\mathcal{M}(I')$). Let $a^\dagger$ be an agent in A' such that $e'(a^\dagger) = e(a^\dagger)$. We need to prove that $u(\mu, d, a^\dagger) \leq u(\mu', d, a^\dagger)$. Let λ_1 (resp., λ'_1) denote brkpts$(I, 1)$ (resp., brkpts$(I', 1)$), and let A_1 (resp., A'_1) denote agents$(I, 1)$ (resp., agents$(I', 1)$). We consider two cases.

Case 1: $a^\dagger \in A_1$. Since $a^\dagger$ belongs to A_1, Lemma 6.3.10 implies that $u(\mu, d, a^\dagger) = \lambda_1 e(a^\dagger)$. The definition of λ'_1 implies that $u(\mu', d, a^\dagger) \geq \lambda'_1 e(a^\dagger)$. Since $u(\mu, d, a^\dagger) = \lambda_1 e(a^\dagger)$ and $u(\mu', d, a^\dagger) \geq \lambda'_1 e(a^\dagger)$, it is sufficient to prove that $\lambda'_1 \geq \lambda_1$. Let $\mu_{A'}$ denote the restriction of μ to A'; thus $\mu_{A'}$ belongs to

frugal(I'). Since μ' belongs to LMMF(I'), we deduce that $\mathbf{u}(I', \mu')$ is lexicographically at least $\mathbf{u}(I', \mu_{A'})$. Hence $\lambda'_1 \geq \lambda_1$, as required.

Case 2: $a^\dagger \in A' \setminus A_1$. Let $\hat{I} = (\hat{A}, B, \hat{e}, \hat{s}, \hat{d})$ and $\hat{I}' = (\hat{A}', B, \hat{e}', \hat{s}', \hat{d}')$ denote the OAFD instances such that $\hat{I} = \text{sub}(I, A_1 \cup (A \setminus A'), \mu)$ and $\hat{I}' = \text{sub}(I', A_1 \cap A', \mu')$. Notice that $\hat{A} = A \setminus (A_1 \cup (A \setminus A')) = A' \setminus A_1 = \hat{A}'$ and $\hat{d} = d_{\hat{A}} = d_{\hat{A}'} = \hat{d}'$. Moreover, the case assumption implies that $a^\dagger$ belongs to $A' \setminus A_1 = \hat{A}$. Let $\hat{\mu}$ and $\hat{\mu}'$ be allocations in $\mathcal{M}(\hat{I})$ and $\mathcal{M}(\hat{I}')$, respectively. By Lemma 6.2.1, it is sufficient to prove that $u(\hat{\mu}, d, a^\dagger) \leq u(\hat{\mu}', d, a^\dagger)$.

Since $\hat{e}$ (resp., $\hat{e}'$) is the restriction of e (resp., e') to $\hat{A}$, we have $\hat{e}'(a) \leq \hat{e}(a)$ for all agents a in $\hat{A}$. Let $\hat{I}^*$ denote the OAFD instance $(\hat{A}, B, \hat{e}, \hat{s}', \hat{d})$, which belongs to shrink($\hat{I}'$). Let $\hat{\mu}^*$ denote an allocation in $\mathcal{M}(\hat{I}^*)$. The induction hypothesis implies that $u(\hat{\mu}', d, a) \geq u(\hat{\mu}^*, d, a)$ for all agents a in $\hat{A}$ such that $\hat{e}(a) = \hat{e}'(a)$. Since $a^\dagger$ belongs to $\hat{A}$ and $\hat{e}'(a^\dagger) = \hat{e}(a^\dagger)$, we deduce that $u(\hat{\mu}^*, d, a^\dagger) \leq u(\hat{\mu}', d, a^\dagger)$. Below we complete the proof by showing that $u(\hat{\mu}, d, a^\dagger) \leq u(\hat{\mu}^*, d, a^\dagger)$.

Let b be an object in B. We have

$$\begin{aligned}
\mu(A_1 \cup (A \setminus A'), b) &\geq \mu(A_1, b) \\
&= s_I(b) \\
&= \min(s(b), \sum_{a \in A} d(a, b)) \\
&\geq \mu'(A', b) \\
&\geq \mu'(A' \cap A_1, b),
\end{aligned}$$

where the first equality holds by Lemma 6.3.6, the second equality holds by the definition of $s_I(b)$, and the second inequality holds because μ' belongs to frugal(I'). Since $\hat{s}(b) = s(b) - \mu(A_1 \cup (A \setminus A'), b)$ and $\hat{s}'(b) = s(b) - \mu'(A' \cap A_1, b)$, we deduce that $\hat{s}(b) \leq \hat{s}'(b)$. Hence Theorem 6.3.19 implies that $u(\hat{\mu}, d, a^\dagger) \leq u(\hat{\mu}^*, d, a^\dagger)$, as required. □

We now turn our attention to proving that any frugal LMMF mechanism is GSP. Bogomolnaia and Moulin [38] prove this result for the special case where all of the demands are either 0 or 1. Like their mechanism, our mechanism partitions the agents into those receiving the minimum utility (i.e., agents($I, 1$)) and the rest. Both proofs proceed to divide the original problem into two subproblems over these two agent subsets. The main difference between these two proofs is that when all of the demands are either 0 or 1, the objects demanded by agents in agents($I, 1$) are no longer available to the remaining agents, and hence there is a clean partitioning of the objects into the subset demanded by agents in agents($I, 1$) and the remaining objects. However, in our setting, the objects fractionally demanded by agents in agents($I, 1$) may still be partly available for the remaining agents, which allows for a more complicated interplay between the two subproblems. We use some new ideas to cope with this added complexity. For example, in the main case (Case 4) of our proof of Theorem 6.3.21, we find it convenient to leverage the RM property established in Theorem 6.3.19. We remark that Bogomolnaia and Moulin also establish RM in their setting, but do not use RM to establish GSP.

Theorem 6.3.21. *Any frugal LMMF mechanism is GSP.*

Proof. The definition of mechanism $\mathcal{M}$ implies that it is sufficient to show that mechanism $\mathcal{M}$ is GSP. For any OAFD instances $I = (A, B, e, s, d)$ and $I' = (A, B, e, s, d')$, any subset A' of A such that $d_{A \setminus A'} = d'_{A \setminus A'}$, and any allocation μ' in $\mathcal{M}(I')$, we define (I, I', A', μ') as a manipulation. For any manipulation $\Phi = (I, I', A', \mu')$ where $I = (A, B, e, s, d)$, we define the set of winning agents, denoted $W(\Phi)$, as $\{a \in A \mid u(\mu', d, a) > e(a)\Lambda_I(a)\}$. Similarly, we define the set $L(\Phi)$ of losing agents as $\{a \in A \mid u(\mu', d, a) < e(a)\Lambda_I(a)\}$. Remark: Lemma 6.3.10 implies that $u(\mu, d, a) = e(a)\Lambda_I(a)$ for all allocations μ in $\mathcal{M}(I)$ and all agents a in A.

Let $P(k)$ denote the predicate "for any manipulation $\Phi = (I, I', A', \mu')$ where $I = (A, B, e, s, d)$, $|A| = k$, and $W(\Phi) \cap A' \neq \emptyset$, we have $L(\Phi) \cap A' \neq \emptyset$." Below we prove by induction on k that $P(k)$ holds for all $k \geq 0$; the claim of the theorem follows immediately.

It is easy to see that $P(0)$ holds. Let k be a positive integer and assume that $P(i)$ holds for $0 \leq i < k$. We need to prove that $P(k)$ holds. Let $\Phi = (I, I', A', \mu')$ be a manipulation where $I = (A, B, e, s, d)$, $I' = (A, B, e, s, d')$, $|A| = k$, and $W(\Phi) \cap A' \neq \emptyset$. We need to prove that $L(\Phi) \cap A' \neq \emptyset$. Let λ_1 (resp., λ'_1) denote brkpts$(I, 1)$ (resp., brkpts$(I', 1)$), and let A_1 (resp., A'_1) denote agents$(I, 1)$ (resp., agents$(I', 1)$). We consider four cases.

Case 1: $\lambda'_1 < \lambda_1$. Lemma 6.3.6 implies that $\lambda'_1 = \text{cap}(I', A'_1)/e(A'_1)$ and $\lambda_1 = \min_{X \subseteq A} \text{cap}(I, X)/e(X)$. Thus

$$\text{cap}(I', A'_1)/e(A'_1) < \min_{X \subseteq A} \text{cap}(I, X)/e(X) \leq \text{cap}(I, A'_1)/e(A'_1),$$

where the first inequality follows from the case assumption and the second inequality follows from $A'_1 \subseteq A$. Multiplying by $e(A'_1)$, we obtain $\mathrm{cap}(I', A'_1) < \mathrm{cap}(I, A'_1)$. If $A' \cap A'_1 = \emptyset$, then $d'_a = d_a$ for all agents a in A'_1 and hence $\mathrm{cap}(I', A'_1) = \mathrm{cap}(I, A'_1)$, a contradiction. It remains to consider the case where $A' \cap A'_1 \neq \emptyset$. Let a belong to $A' \cap A'_1$. Thus

$$u(\mu', d, a) \leq \mu'(a, B) = u(\mu', d', a) = e(a)\lambda'_1 < e(a)\lambda_1,$$

where the two equalities follows from Lemma 6.3.10. Hence a is in $L(\Phi) \cap A'$.

Case 2: $\lambda'_1 \geq \lambda_1$ and $L(\Phi) \cap A_1 \neq \emptyset$. Let a be an agent in $L(\Phi) \cap A_1$. If a is in A' then $L(\Phi) \cap A' \neq \emptyset$, as required. Thus, in what follows, we assume that a is not in A'. Let i denote the least integer such that a is in agents(I', i). We have

$$e(a)\Lambda_{I'}(a) = u(\mu', d', a) = u(\mu', d, a) < e(a)\lambda(a) = e(a)\lambda_1,$$

where the first equality holds by Lemma 6.3.10, the second equality holds since a is not in A' and hence $d'_a = d_a$, the inequality holds since a is in $L(\Phi)$, and the third equality holds since a is in A_1. Thus $\lambda'_1 \leq \Lambda_{I'}(a) < \lambda_1$, contradicting the first condition in the case assumption.

Case 3: $\lambda'_1 \geq \lambda_1$, $L(\Phi) \cap A_1 = \emptyset$, and $W(\Phi) \cap A_1 \neq \emptyset$. Let a denote an agent in $W(\Phi) \cap A_1$. Thus $u(\mu', d, a) > e(a)\Lambda_I(a) = e(a)\lambda_1$. Since $u(\mu', d, A_1) \leq \mathrm{cap}(I, A_1) = e(A_1)\lambda_1$ by Lemma 6.3.6, we deduce that $u(\mu', d, A_1 - a) < e(A_1 - a)\lambda_1$. Thus there is an agent a' in $A_1 - a$ such that $u(\mu', d, a') < e(a')\lambda_1 = e(a')\Lambda_I(a')$. Hence $L(\Phi) \cap A_1 \neq \emptyset$, contradicting the second condition in the case assumption.

Case 4: $\lambda_1' \geq \lambda_1$, $L(\Phi) \cap A_1 = \emptyset$, and $W(\Phi) \cap A_1 = \emptyset$. Let μ denote an allocation in $\mathcal{M}(I)$. Let $\hat{I} = (\hat{A}, B, \hat{e}, \hat{s}, \hat{d})$ denote the OAFD instance $\text{sub}(I, A_1, \mu)$; thus $\hat{A} = A \setminus A_1$. Let $\hat{\mu}$ denote the restriction of μ to $\hat{A}$; Lemma 6.2.1, implies that $\hat{\mu}$ is in $\mathcal{M}(\hat{I})$. Let $I^* = (\hat{A}, B, \hat{e}, s^*, d^*)$ denote the OAFD instance $\text{sub}(I, A_1, \mu')$ and let μ^* denote the restriction of μ' to $\hat{A}$; Lemma 6.2.1, implies that μ^* is in $\mathcal{M}(I^*)$. Let $\tilde{I}$ denote the OAFD instance $(\hat{A}, B, \hat{e}, s^*, \hat{d})$ and let $\tilde{\mu}$ be in $\mathcal{M}(\tilde{I})$.

Claim 1: $\Lambda_I(a) \geq \Lambda_{\tilde{I}}(a)$ holds for all agents a in $\hat{A}$. The third condition in the case assumption implies that $u(\mu', d, a) \geq e(a)\lambda_1$ for all agents a in A_1. Thus Lemma 6.3.13 implies that $\mu(A_1, b) \leq \mu'(A_1, b)$ for all objects b in B. It follows that $\hat{s}(b) \geq s^*(b)$ for all objects b in B. Hence

$$e(a)\Lambda_I(a) = u(\mu, d, a) = u(\hat{\mu}, \hat{d}, a) \geq u(\tilde{\mu}, \hat{d}, a) = e(a)\Lambda_{\tilde{I}}(a),$$

where the first and last equalities hold by Lemma 6.3.10, the second equality holds by the definition of $\hat{\mu}$, and the inequality holds by Theorem 6.3.19. Dividing by $e(a)$ yields the claim.

Claim 2: $u(\mu', d, a) = u(\mu^*, \hat{d}, a)$ for all agents a in $\hat{A}$. We have

$$u(\mu', d, a) = \sum_{b \in B} \min(\mu'(a, b), d(a, b)) = \sum_{b \in B} \min(\mu^*(a, b), \hat{d}(a, b)) = u(\mu^*, \hat{d}, a),$$

where the second equality holds by the definition of μ^*. The claim follows.

Let A'' denote $A' \setminus A_1$ and let Φ' denote the manipulation $(\tilde{I}, I^*, A'', \mu^*)$.

Claim 3: $W(\Phi') \cap A'' \neq \emptyset$. Since $W(\Phi) \cap A' \neq \emptyset$, the third condition in the case assumption implies that $W(\Phi) \cap A'' \neq \emptyset$. Let a be an agent in

$W(\Phi) \cap A''$. Thus

$$u(\mu^*, \hat{d}, a) = u(\mu', d, a) > e(a)\Lambda_I(a) \geq e(a)\Lambda_{\tilde{I}}(a),$$

where the first equality holds by Claim 2, the first inequality holds because a is in $W(\Phi)$, and the second inequality holds by Claim 1. Since $u(\mu^*, \hat{d}, a) > \Lambda_{\tilde{I}}(a)$ and a is in A'', the claim holds.

Since $|\hat{A}| < k$ and Claim 3 holds, the induction hypothesis implies that $L(\Phi') \cap A'' \neq \emptyset$. Let a be in $L(\Phi') \cap A''$. Thus a is in $A'' \subseteq \hat{A}$ and

$$u(\mu', d, a) = u(\mu^*, \hat{d}, a) < e(a)\Lambda_{\tilde{I}}(a) \leq e(a)\Lambda_I(a),$$

where the equality holds by Claim 2, the first inequality holds because a is in $L(\Phi')$, and the second inequality holds by Claim 1. Since $u(\mu', d, a) < e(a)\Lambda_I(a)$ and a is in $A'' \subseteq A'$, we deduce that a is in $W(\Phi) \cap A'$. □

6.4 Impossibility Results

In this section, we show that fairness and SI are incompatible properties. Lemma 6.4.1 below establishes that for any $\alpha > 1/2$, no OAFD mechanism is α-SI and MMF. As mentioned in Section 6.2, MMF is a weaker notion of fairness than LMMF. Thus for any $\alpha > 1/2$, no OAFD mechanism is α-SI and LMMF.

Lemma 6.4.1. *For any $\alpha > 1/2$, no* OAFD *mechanism is α-SI and MMF.*

Proof. Let M be an MMF OAFD mechanism. Consider an OAFD instance $I = (A, B, e, s, d)$ with n agents $a_1, \ldots, a_n$, each with endowment 1, and two

objects b_1 and b_2, each with supply n, and where $d(a_1, b_1) = d(a_1, b_2) = 1$, and $d(a, b_1) = 2$ and $d(a, b_2) = 0$ for all agents a in $A - a_1$. Mechanism M gives a utility of $1 + 1/n$ to each agent in A. If agent a_1 is allocated an $e(a_1)/e(A) = 1/n$ fraction of each object, then a_1 achieves utility 2. Hence M is at most $\frac{1}{2}(1 + \frac{1}{n})$-SI. Let α be any value greater than $1/2$. Then, choosing a sufficiently large n, we find that M is not α-SI. □

Since no mechanism can be MMF and SI, we consider the following natural relaxation: mechanisms that are MMF subject to being SI. Formally, for any OAFD instance $I = (A, B, e, s, d)$, we say that an allocation μ in allocs(I) is MMF-SI if μ maximizes $\min_{a \in A} u(\mu', d, a)/e(a)$ over all μ' in allocs(I) such that μ' is SI. An OAFD mechanism M is MMF-SI if for any OAFD instance I, every allocation in $M(I)$ is MMF-SI. Lemma 6.4.2 below shows that the SP and MMF-SI properties are incompatible.

Lemma 6.4.2. *No* OAFD *mechanism is SP and MMF-SI.*

Proof. Let M be an MMF-SI OAFD mechanism. Consider an OAFD instance $I = (A, B, e, s, d)$ with three agents a_1, a_2, and a_3, each with endowment 1, and two objects b_1 and b_2, each with supply 6, and where $d(a_1, b_1) = 3$, $d(a_1, b_2) = 1$, $d(a_2, b_1) = d(a_3, b_1) = 0$, and $d(a_2, b_2) = d(a_3, b_2) = 3$. Let μ belong to $M(I)$. It is easy to verify that $u(\mu, d, a_1) = 3$. Let d' denote (d_{A-a_1}, d''), where d'' belongs to demands($\{a_1\}, B$), $d''(a_1, b_1) = 3$, and $d''(a_1, b_2) = 2$. Let I' denote the OAFD instance (A, B, e, s, d') and let μ' belong to $M(I')$. It is easy to verify that $u(\mu', d', a_1) = 4$. We conclude that mechanism M is not SP. □

Chapter 7

Open Problems

This chapter summarizes open problems related to previous chapters and may be of independent interests.

In Chapter 2, we have introduced the agent-moving model, and presented a polynomial-time algorithm for finding Pareto-efficient matchings on generalized stars in the agent-moving model, a problem that remains open in the object-moving model. It is natural to ask whether our techniques can be extended to address this open problem. Our algorithm relies on the polynomial-time solvability of the reachable object problem for the center agent, which allows us to compute an object that is matched to the center agent in some Pareto-efficient matching. In the object-moving model, no polynomial-time algorithm is known to compute an agent that is matched to the center object in some Pareto-efficient matching. (We do know how to compute the agents that can be reached by the center object in polynomial time, but it isn't clear how to use this information to compute a Pareto-efficient matching in polynomial time.)

As in the work presented in Chapter 3, it is interesting to search for Pareto-efficient matchings satisfying other desiderata. In Chapter 3, we have

explored such a refinement in the direction of popularity. Our refinement seeks to maximize the number of agents who achieve a better outcome than in the initial configuration, and we show that this refinement is NP-hard to achieve. Our complexity proofs show that even for trees or stars with ties, determining a maximum votes Pareto-efficient (MVPE) matching is NP-hard. It is interesting to investigate the complexity of this refinement in the context of other markets with an initial configuration. A challenging open question is to completely characterize the social network structures for which case the MVPE problem is polynomial-time solvable. Another direction is to consider a matching that maximizes the total happiness of all agents. At the same time, it is possible that few agents are happy in a matching that maximizes total happiness, so it may be important to incorporate a fairness criterion as well.

The NP-hardness results presented in Chapter 4 suggest the need to consider approximation algorithms or heuristic algorithms. As a starting point, one could study the approximation algorithms for finding social welfare maximizing k-partition in FHGs played on undirected unweighted graphs. Note that for this problem, it is easy to see that the classical algorithm finding the densest subgraph by Goldberg [90] provides a k-approximation algorithm. It is interesting to study whether there is an efficient $O(\log k)$-approximation algorithm. Furthermore, it is interesting to study the price of anarchy and price of stability for Nash stable partitions in our model, in particular, study the performance in terms of k. Finally, we conjecture that it is NP-hard to find

a Nash stable k-partition on undirected unweighted graphs for all $k \geq 2$, but this remains an open problem.

In Chapter 5, we studied the obnoxious facility location game with dichotomous preferences. This game generalizes obnoxious facility location game to more realistic scenarios. All of the mechanisms presented in this chapter run in polynomial time, except that the running time of Mechanism 2 has exponential dependence on k (and polynomial dependence on n). We showed that Mechanism 2 is WGSP for all k and is efficient for $k \leq 3$. Properties 1 and 2 in the proof of our associated theorem, Theorem 5.5.4, do not hold for $k > 3$. Nevertheless, we conjecture that Mechanism 2 is efficient for all k. It remains an interesting open problem to reduce the gap between the $\Omega(\sqrt{n})$ and $O(n)$ bounds on the approximation ratio α of WGSP α-egalitarian mechanisms.

In Chapter 6, we propose three possible directions. First, we have shown that our mechanism is α-SI for $\alpha = 1/2$, but in most real world instances it might achieve α-SI for a significantly higher value of α. It would be interesting to benchmark our mechanism on real data. Second, our work assumes perfect knowledge of future demands. We want to develop mechanisms whose performance degrades gracefully as the knowledge of future demands becomes more unreliable. Finally, we have studied lexicographic maximin fairness in this chapter. It would also be interesting to study other notions of fairness.

Bibliography

[1] Atila Abdulkadiroğlu, Yeon-Koo Che, and Yosuke Yasuda. Resolving conflicting preferences in school choice: The "Boston Mechanism" reconsidered. *American Economic Review*, 101(1):399–410, 2011.

[2] Atila Abdulkadiroğlu, Parag A. Pathak, and Alvin E. Roth. The New York City high school match. *American Economic Review*, 95(2):364–367, 2005.

[3] Atila Abdulkadiroğlu and Tayfun Sönmez. Random serial dictatorship and the core from random endowments in house allocation problems. *Econometrica*, 66(3):689–701, 1998.

[4] Atila Abdulkadiroğlu and Tayfun Sönmez. House allocation with existing tenants. *Journal of Economic Theory*, 88(2):233–260, 1999.

[5] David J. Abraham, Katarína Cechlárová, David F. Manlove, and Kurt Mehlhorn. Pareto optimality in house allocation problems. In *Proceedings of the 15th International Symposium on Algorithms and Computation*, pages 3–15, 2004.

[6] David J. Abraham, Robert W. Irving, Telikepalli Kavitha, and Kurt Mehlhorn. Popular matchings. *SIAM Journal on Computing*, 37(4):1030–1045, 2007.

[7] Aishwarya Agarwal, Edith Elkind, Jiarui Gan, and Alexandros A. Voudouris. Swap stability in schelling games on graphs. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, pages 1758–1765, 2020.

[8] Martin Aleksandrov, Haris Aziz, Serge Gaspers, and Toby Walsh. Online fair division: Analysing a food bank problem. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 2540–2546, 2015.

[9] Martin Aleksandrov and Toby Walsh. Pure Nash equilibria in online fair division. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 42–48, 2017.

[10] Noga Alon, Michal Feldman, Ariel D. Procaccia, and Moshe Tennenholtz. Strategyproof approximation of the minimax on networks. *Mathematics of Operations Research*, 35(3):513–526, 2010.

[11] Zhao An, Qilong Feng, Iyad Kanj, and Ge Xia. The complexity of tree partitioning. *Algorithmica*, 82(9):2606–2643, 2020.

[12] Eleftherios Anastasiadis and Argyrios Deligkas. Heterogeneous facility location games. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 623–631, 2018.

[13] Kenneth J. Arrow. *Social Choice and Individual Values*. Wiley, 1951.

[14] Kenneth J. Arrow, Amartya K. Sen, and Kotaro Suzumura, editors. *Handbook of Social Choice and Welfare*, volume 1. Elsevier, 2002.

[15] Kenneth J. Arrow, Amartya K. Sen, and Kotaro Suzumura, editors. *Handbook of Social Choice and Welfare*, volume 2. Elsevier, 2010.

[16] Haris Aziz, Páter Biró, Jérôme Lang, Julien Lesca, and Jérôme Monnot. Optimal reallocation under additive and ordinal preferences. In *Proceedings of the 15th International Conference on Autonomous Agents and MultiAgent Systems*, pages 402–410, 2016.

[17] Haris Aziz, Florian Brandl, Felix Brandt, Paul Harrenstein, Martin Olsen, and Dominik Peters. Fractional hedonic games. *ACM Transactions on Economics and Computation*, 7(2):6:1–6:29, 2019.

[18] Haris Aziz, Felix Brandt, and Hans Georg Seedig. Computing desirable partitions in additively separable hedonic games. *Artificial Intelligence*, 195:316–334, 2013.

[19] Haris Aziz and Bart De Keijzer. Housing markets with indifferences: A tale of two mechanisms. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, pages 1249–1255, 2012.

[20] Haris Aziz, Serge Gaspers, Joachim Gudmundsson, Julián Mestre, and Hanjo Täubig. Welfare maximization in fractional hedonic games. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, pages 461–467, 2015.

[21] Haris Aziz, Serge Gaspers, Simon Mackenzie, and Toby Walsh. Fair assignment of indivisible objects under ordinal preferences. *Artificial*

Intelligence, 227:71–92, 2015.

[22] Haris Aziz and Rahul Savani. Hedonic games. In Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D. Procaccia, editors, *Handbook of Computational Social Choice*, pages 356–376. Cambridge University Press, 2016.

[23] Cristina Bazgan, Janka Chlebíková, and Clément Dallard. Graphs without a partition into two proportionally dense subgraphs. *Information Processing Letters*, 155:105877, 2020.

[24] Cristina Bazgan, Janka Chlebikova, and Thomas Pontoizeau. Structural and algorithmic properties of 2-community structures. *Algorithmica*, 80(6):1890–1908, 2018.

[25] Cristina Bazgan, Zsolt Tuza, and Daniel Vanderpooten. Complexity and approximation of satisfactory partition problems. In *Proceedings of the 11th International Computing and Combinatorics Conference*, pages 829–838, 2005.

[26] Cristina Bazgan, Zsolt Tuza, and Daniel Vanderpooten. The satisfactory partition problem. *Discrete Applied Mathematics*, 154(8):1236–1245, 2006.

[27] Fred A. Behringer. A simplex based algorithm for the lexicographically extended linear maxmin problem. *European Journal of Operational Research*, 7(3):274–283, 1981.

[28] Matthias Bentert, Jiehua Chen, Vincent Froese, and Gerhard Woeginger. Good things come to those who swap objects on paths. *arXiv:1905.04219*, 2019.

[29] Dimitri P. Bertsekas, Robert G. Gallager, and Pierre Humblet. *Data Networks*. Prentice-Hall, Englewood Cliffs, 1992.

[30] Aurélie Beynier, Yann Chevaleyre, Laurent Gourvès, Ararat Harutyunyan, Julien Lesca, Nicolas Maudet, and Anaëlle Wilczynski. Local envy-freeness in house allocation problems. *Autonomous Agents and Multi-Agent Systems*, 33(5):591–627, 2019.

[31] Aurélie Beynier, Nicolas Maudet, Simon Rey, and Parham Shams. An optimal procedure to check pareto-optimality in house markets with single-peaked preferences. *arXiv:2002.11660*, 2020.

[32] Davide Bilò, Vittorio Bilò, Pascal Lenzner, and Louise Molitor. Topological influence and locality in swap schelling games. In *Proceedings of 45th International Symposium on Mathematical Foundations of Computer Science*, pages 15:1–15:15, 2020.

[33] Vittorio Bilò, Ioannis Caragiannis, Michele Flammini, Ayumi Igarashi, Gianpiero Monaco, Dominik Peters, Cosimo Vinci, and William S. Zwicker. Almost envy-free allocations with connected bundles. In *Proceedings of the 10th Innovations in Theoretical Computer Science Conference*, pages 14:1–14:21, 2018.

[34] Vittorio Bilò, Angelo Fanelli, Michele Flammini, Gianpiero Monaco, and Luca Moscardelli. Nash stability in fractional hedonic games. In *Proceedings of the 10th International Workshop on Internet and Network Economics*, pages 486–491, 2014.

[35] Péter Biró and Jens Gudmundsson. Complexity of finding Pareto-efficient allocations of highest welfare. *European Journal of Operational Research*, 291(2):614–628, 2021.

[36] Péter Biró, David F Manlove, and Shubham Mittal. Size versus stability in the marriage problem. *Theoretical Computer Science*, 411(16-18):1828–1841, 2010.

[37] Anna Bogomolnaia, Matthew O Jackson, et al. The stability of hedonic coalition structures. *Games and Economic Behavior*, 38(2):201–230, 2002.

[38] Anna Bogomolnaia and Hervé Moulin. Random matching under dichotomous preferences. *Econometrica*, 72(1):257–279, 2004.

[39] Paul S. Bonsma, Hajo Broersma, Viresh Patel, and Artem V. Pyatkin. The complexity of finding uniform sparsest cuts in various graph classes. *Journal of Discrete Algorithms*, 14:136–149, 2012.

[40] Eric Boutin, Jaliya Ekanayake, Wei Lin, Bing Shi, Jingren Zhou, Zhengping Qian, Ming Wu, and Lidong Zhou. Apollo: Scalable and coordinated scheduling for cloud-scale computing. In *Proceedings of the 11th*

USENIX Conference on Operating Systems Design and Implementation, pages 285–300, 2014.

[41] Steven J. Brams and Peter C. Fishburn. Approval voting. *The American Political Science Review*, 72(3):831–847, 1978.

[42] Ulrik Brandes. *Network Analysis: Methodological Foundations*. Springer Science & Business Media, 2005.

[43] Florian Brandl, Felix Brandt, and Martin Strobel. Fractional hedonic games: Individual and group stability. In *Proceedings of the 14th International Conference on Autonomous Agents and Multi-Agent Systems*, pages 1219–1227, 2015.

[44] Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D. Procaccia, editors. *Handbook of Computational Social Choice*. Cambridge University Press, 2016.

[45] J. Randall Brown. The sharing problem. *Operations Research*, 27(2):324–340, 1979.

[46] Martin Bullinger. Pareto-optimality in cardinal hedonic games. In *Proceedings of the 19th International Conference on Autonomous Agents and Multi-Agent Systems*, pages 213–221, 2020.

[47] Aydın Buluç, Henning Meyerhenke, Ilya Safro, Peter Sanders, and Christian Schulz. Recent advances in graph partitioning. In *Algorithm Engineering*, pages 117–158. Springer, 2016.

[48] Raffaello Carosi, Gianpiero Monaco, and Luca Moscardelli. Local core stability in simple symmetric fractional hedonic games. In *Proceedings of the 18th International Conference on Autonomous Agents and Multi-Agent Systems*, pages 574–582, 2019.

[49] Georgios Chalkiadakis, Edith Elkind, and Michael Wooldridge. Computational aspects of cooperative game theory. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 5(6):1–168, 2011.

[50] Ray M. Chang, Robert J. Kauffman, and Kwon Youngok. Understanding the paradigm shift to computational social science in the presence of big data. *Decision Support Systems*, 63:67–80, 2014.

[51] Siyuan Chen, Wei Liu, Jiamou Liu, Khí-Uí Soo, and Wu Chen. Maximizing social welfare in fractional hedonic games using shapley value. In *Proceedings of the 4th IEEE International Conference on Agents*, pages 21–26, 2019.

[52] Wei Chen, Zhenming Liu, Xiaorui Sun, and Yajun Wang. A game-theoretic framework to identify overlapping communities in social networks. *Data Mining and Knowledge Discovery*, 21(2):224–240, 2010.

[53] Zhihuai Chen, Ken C. K. Fong, Minming Li, Kai Wang, Hongning Yuan, and Yong Zhang. Facility location games with optional preference. *Theoretical Computer Science*, 847:185–197, 2020.

[54] Yukun Cheng, Qiaoming Han, Wei Yu, and Guochuan Zhang. Obnoxious facility game with a bounded service range. In *Proceedings of the 10th Annual Conference on Theory and Applications of Models of Computation*, pages 272–281, 2013.

[55] Yukun Cheng, Wei Yu, and Guochuan Zhang. Mechanisms for obnoxious facility game on a path. In *Proceedings of the 5th International Conference on Combinatorial Optimization and Applications*, pages 262–271, 2011.

[56] Yukun Cheng, Wei Yu, and Guochuan Zhang. Strategy-proof approximation mechanisms for an obnoxious facility game on networks. *Theoretical Computer Science*, 497:154–163, 2013.

[57] Yann Chevaleyre, Ulle Endriss, and Nicolas Maudet. Allocating goods on a graph to eliminate envy. In *Proceedings of the 22nd National Conference on Artificial Intelligence*, pages 700–705, 2007.

[58] Edward H. Clarke. Multipart pricing of public goods. *Public Choice*, 11(1):17–33, 1971.

[59] Richard Cole, Vasilis Gkatzelis, and Gagan Goel. Mechanism design for fair division: Allocating divisible items without payments. In *Proceedings of the 14th ACM Conference on Electronic Commerce*, pages 251–268, 2013.

[60] Vincent Conitzer and Rupert Freeman. Algorithmically driven shared ownership economies. pages 275–285, 2019.

[61] Ágnes Cseh. Popular matchings. In Ulle Endriss, editor, *Trends in Computational Social Choice*, chapter 6, pages 105–122. AI Access, 2017.

[62] Anastasia Damamme, Aurélie Beynier, Yann Chevaleyre, and Nicolas Maudet. The power of swap deals in distributed resource allocation. In *Proceedings of the 14th International Conference on Autonomous Agents and MultiAgent Systems*, pages 625–633, 2015.

[63] Julien Darlay, Nadia Brauner, and Julien Moncel. Dense and sparse graph partition. *Discrete Applied Mathematics*, 160(16-17):2389–2396, 2012.

[64] John P. Dickerson, Ariel D. Procaccia, and Tuomas Sandholm. Optimizing kidney exchange with transplant chains: Theory and reality. In *Proceedings of the 11th International Conference on Autonomous Agents and MultiAgent Systems*, pages 711–718, 2012.

[65] Elad Dokow, Michal Feldman, Reshef Meir, and Ilan Nehama. Mechanism design on discrete lines and cycles. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, pages 423–440, 2012.

[66] Riccardo Dondi and Danny Hermelin. Computing the k densest subgraphs of a graph. *arXiv:2002.07695*, 2020.

[67] Riccardo Dondi, Mohammad Mehdi Hosseinzadeh, Giancarlo Mauri, and Italo Zoppis. Top-k overlapping densest subgraphs: Approximation and complexity. In *Proceedings of the 20th Italian Conference on Theoretical Computer Science*, pages 110–121, 2019.

[68] Melvin Dresher. *Games of Strategy: Theory and Applications*. Prentice-Hall, Englewood Cliffs, 1961.

[69] Jacques H. Dreze and Joseph Greenberg. Hedonic coalitions: Optimality and stability. *Econometrica: Journal of the Econometric Society*, pages 987–1003, 1980.

[70] Lingjie Duan, Bo Li, Minming Li, and Xinping Xu. Heterogeneous two-facility location games with minimum distance requirement. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1461–1469, 2019.

[71] Federico Echenique, Antonio Miralles, and Jun Zhang. Fairness and efficiency for probabilistic allocations with endowments. *arXiv:1908.04336*, 2019.

[72] Mark J. Eisner and Dennis G. Severance. Mathematical techniques for efficient record segmentation in large shared databases. *Journal of the ACM*, 23(4):619–635, 1976.

[73] Ulrich Endriss, Nicolas Maudet, Fariba Sadri, and Francesca Toni. Negotiating socially optimal allocations of resources. *Journal of Artificial*

Intelligence Research, 25:315–348, 2006.

[74] Vladmir Estivill-Castro and Mahdi Parsa. On connected two communities. In *Proceedings of the 36th Australasian Computer Science Conference*, pages 23–30, 2013.

[75] Michal Feldman and Yoav Wilf. Strategyproof facility location and the least squares objective. In *Proceedings of the 14th ACM Conference on Electronic Commerce*, pages 873–890, 2013.

[76] Aris Filos-Ratsikas, Minming Li, Jie Zhang, and Qiang Zhang. Facility location with double-peaked preferences. *Autonomous Agents and Multi-Agent Systems*, 31(6):1209–1235, 2017.

[77] Michele Flammini, Gianpiero Monaco, Luca Moscardelli, Mordechai Shalom, and Shmuel Zaks. Online coalition structure generation in graph games. In *Proceedings of the 17th International Conference on Autonomous Agents and Multi-Agent Systems*, pages 1353–1361, 2018.

[78] Michele Flammini, Gianpiero Monaco, and Qiang Zhang. Strategyproof mechanisms for additively separable hedonic games and fractional hedonic games. In *Proceedings of the 15th International Workshop on Approximation and Online Algorithms*, pages 301–316, 2017.

[79] Lester R. Ford and Delbert R. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.

[80] Santo Fortunato and Darko Hric. Community detection in networks: A user guide. *Physics Reports*, 659:1–44, 2016.

[81] Dimitris Fotakis and Christos Tzamos. Winner-imposing strategyproof mechanisms for multiple facility location games. In *Proceedings of the 6th International Workshop on Internet and Network Economics*, pages 234–245, 2010.

[82] Dimitris Fotakis and Christos Tzamos. Strategyproof facility location with concave costs. *SIGecom Exchanges*, 12(2):46–49, 2014.

[83] Rupert Freeman, Seyed Majid Zahedi, Vincent Conitzer, and Benjamin C. Lee. Dynamic proportional sharing: A game-theoretic approach. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 2(1):1–36, 2018.

[84] Yuhei Fukui, Aleksandar Shurbevski, and Hiroshi Nagamochi. λ-group strategy-proof mechanisms for the obnoxious facility game in star networks. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E102.A:1179–1186, 2019.

[85] Giorgio Gallo, Michael D. Grigoriadis, and Robert E. Tarjan. A fast parametric maximum flow algorithm and applications. *SIAM Journal on Computing*, 18(1):30–55, 1989.

[86] Peter Gardenfors. Assignment problem based on ordinal preferences. *Management Science*, 20(3):331–340, 1973.

[87] Michael U. Gerber and Daniel Kobler. Classes of graphs that can be partitioned to satisfy all their vertices. *Australasian Journal of Combinatorics*, 29:201–214, 2004.

[88] Ali Ghodsi, Matei Zaharia, Benjamin Hindman, Andy Konwinski, Scott Shenker, and Ion Stoica. Dominant resource fairness: Fair allocation of multiple resource types. In *Proceedings of the 8th USENIX Symposium on Networked Systems Design and Implementation*, pages 323–336, 2011.

[89] Ali Ghodsi, Matei Zaharia, Scott Shenker, and Ion Stoica. Choosy: Max-min fair sharing for datacenter jobs with constraints. In *Proceedings of the 8th ACM European Conference on Computer Systems*, pages 365–378, 2013.

[90] Andrew V. Goldberg. Finding a maximum density subgraph. Technical report, University of California Berkeley, 1984.

[91] Laurent Gourvès, Julien Lesca, and Anaëlle Wilczynski. Object allocation via swaps along a social network. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 213–219, 2017.

[92] Theodore Groves. Incentives in teams. *Econometrica*, 41(4):617–631, 1973.

[93] Ridi Hossain. Sharing is caring: Dynamic mechanism for shared resource ownership. In *Proceedings of the 18th International Conference*

on Autonomous Agents and MultiAgent Systems, pages 2417–2419, 2019.

[94] Sen Huang and Mingyu Xiao. Object reachability via swaps along a line. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, pages 2037–2044, 2019.

[95] Sen Huang and Mingyu Xiao. Object reachability via swaps under strict and weak preferences. *Autonomous Agents and Multi-Agent Systems*, 34(2):51, 2020.

[96] Ken Ibara and Hiroshi Nagamochi. Characterizing mechanisms in obnoxious facility game. In *Proceedings of the 6th International Conference Combinatorial Optimization and Applications*, pages 301–311, 2012.

[97] Ayumi Igarashi and Dominik Peters. Pareto-optimal allocation of indivisible goods with connectivity constraints. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, pages 2045–2052, 2019.

[98] Ramin Javadi and Amir Nikabadi. On the parameterized complexity of sparsest cut and small-set expansion problems. *arXiv:1910.12353*, 2019.

[99] Annapurna Jonnalagadda and Lakshmanan Kuppusamy. A survey on game theoretic models for community detection in social networks. *Social Network Analysis and Mining*, 6(1):1–24, 2016.

[100] Christos Kaklamanis, Panagiotis Kanellopoulos, and Konstantinos Papaioannou. The price of stability of simple symmetric fractional hedonic

games. In *Proceedings of 9th International Symposium on Algorithmic Game Theory*, pages 220–232, 2016.

[101] Kirthevasan Kandasamy, Gur-Eyal Sela, Joseph E Gonzalez, Michael I. Jordan, and Ion Stoica. Online learning demands in max-min fairness. *arXiv:2012.08648*, 2020.

[102] Richard M. Karp. Reducibility among combinatorial problems. In *Proceedings of a Symposium on the Complexity of Computer Computations*, pages 85–103, 1972.

[103] Akshay-Kumar Katta and Jay Sethuraman. A solution to the random assignment problem on the full preference domain. *Journal of Economic Theory*, 131(1):231–250, 2006.

[104] Telikepalli Kavitha, Julián Mestre, and Meghana Nasre. Popular mixed matchings. *Theoretical Computer Science*, 412(24):2679–2690, 2011.

[105] Telikepalli Kavitha, Meghana Nasre, and Prajakta Nimbhorkar. Popularity at minimum cost. *Journal of Combinatorial Optimization*, 27(3):574–596, 2014.

[106] Bisma S. Khan and Muaz A. Niazi. Network community detection: A review and visual survey. *arXiv:1708.00977*, 2017.

[107] Samir Khuller and Barna Saha. On finding dense subgraphs. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming*, pages 597–608, 2009.

[108] Bettina Klaus, David Manlove, and Francesca Rossi. *Matching under preferences*. Cambridge University Press, 2016.

[109] Rachelle S. Klein, Hanan Luss, and Uriel G. Rothblum. Multiperiod allocation of substitutable resources. *European Journal of Operational Research*, 85(3):488–503, 1995.

[110] Rachelle S. Klein, Hanan Luss, and Donald R. Smith. A lexicographic minimax algorithm for multiperiod resource allocation. *Mathematical Programming*, 55(1):213–234, 1992.

[111] Piotr Krysta, David Manlove, Baharak Rastegari, and Jinshan Zhang. Size versus truthfulness in the house allocation problem. *Algorithmica*, 81(9):3422–3463, 2019.

[112] Piotr Krysta, David F. Manlove, Baharak Rastegari, and Jinshan Zhang. Size versus truthfulness in the house allocation problem. In *Proceedings of the 14th ACM conference on Economics and Computation*, pages 453–470, 2014.

[113] Jeremy Kun, Brian Powers, and Lev Reyzin. Anti-coordination games and stable graph colorings. In *Proceedings of the 6th International Symposium on Algorithmic Game Theory*, pages 122–133, 2013.

[114] Jure Leskovec, Lada A. Adamic, and Bernardo A. Huberman. The dynamics of viral marketing. *ACM Transactions on the Web*, 1(1):5–es, 2007.

[115] Daniel Leven and Zvi Galil. NP-completeness of finding the chromatic index of regular graphs. *Journal of Algorithms*, 4(1):35–44, 1983.

[116] Fu Li. Fractional hedonic games with a limited number of coalitions. In *Proceedings of the 22nd Italian Conference on Theoretical Computer Science*, pages 205–218, 2021.

[117] Fu Li, C. Gregory Plaxton, and Vaibhav B. Sinha. The obnoxious facility location game with dichotomous preferences. In *Proceedings of the 22nd Italian Conference on Theoretical Computer Science*, pages 219–233. The full version appears in the arXiv:2109.05396.

[118] Fu Li, C. Gregory Plaxton, and Vaibhav B. Sinha. Egalitarian resource sharing over multiple rounds. *arXiv:2106.02688*, 2021.

[119] Fu Li, C. Gregory Plaxton, and Vaibhav B. Sinha. Object allocation over a network of objects: Mobile agents with strict preferences. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1578–1580, 2021. Extended abstract. The full version appears in the arXiv:2103.01394.

[120] Fu Li and Xiong Zheng. Maximum votes Pareto-efficient allocations via swaps on a social network. In *Proceedings of the 46th International Symposium on Mathematical Foundations of Computer Science*, pages 71:1–71:16, 2021.

[121] Minming Li, Pinyan Lu, Yuhao Yao, and Jialin Zhang. Strategyproof mechanism for two heterogeneous facilities with constant approximation ratio. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence*, pages 238–245, 2020.

[122] Pinyan Lu, Xiaorui Sun, Yajun Wang, and Zeyuan Allen Zhu. Asymptotically optimal strategy-proof mechanisms for two-facility games. In *Proceedings of the 11th ACM Conference on Electronic Commerce*, pages 315–324, 2010.

[123] Hanan Luss. On equitable resource allocation problems: A lexicographic minimax approach. *Operations Research*, 47(3):361–378, 1999.

[124] Mohammad Mahdian. Random popular matchings. In *Proceedings of the 7th ACM Conference on Electronic Commerce*, pages 238–242, 2006.

[125] David F. Manlove. *Algorithmics of Matching Under Preferences*. World Scientific, 2013.

[126] David W. Matula and Farhad Shahrokhi. Sparsest cuts and bottlenecks in graphs. *Discrete Applied Mathematics*, 27(1-2):113–123, 1990.

[127] Richard Matthew McCutchen. The least-unpopularity-factor and least-unpopularity-margin criteria for matching problems with one-sided preferences. In *Proceedings of the 8th Latin American Symposium on Theoretical Informatics*, pages 593–604, 2008.

[128] Nimrod Megiddo. Optimal flows in networks with multiple sources and sinks. *Mathematical Programming*, 7(1):97–107, 1974.

[129] Nimrod Megiddo. A good algorithm for lexicographically optimal flows in multi-terminal networks. *Bulletin of the American Mathematical Society*, 83(3):407–409, 1977.

[130] Luis Müller and Matthias Bentert. On reachable assignments in cycles and cliques. *arXiv:2005.02218*, 2020.

[131] Mark E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

[132] Wlodzimierz Ogryczak, Hanan Luss, Michal Pióro, Dritan Nace, and Artur Tomaszewski. Mechanism design for house allocation problems: A short introduction. *Optima*, 82:2–8, 2010.

[133] Wlodzimierz Ogryczak, Hanan Luss, Michal Pióro, Dritan Nace, and Artur Tomaszewski. Fair optimization and networks: A survey. *Journal of Applied Mathematics*, 2014:612018:1–612018:25, 2014.

[134] Wlodzimierz Ogryczak, Michal Pióro, and Artur Tomaszewski. Telecommunications network design and max-min optimization problem. *Journal of Telecommunications and Information Technology*, 4:43–56, 2005.

[135] Włodzimierz Ogryczak and Tomasz Śliwiński. On direct methods for lexicographic min-max optimization. In *Proceedings of the 6th Interna-*

tional Conference on Computational Science and Its Applications, pages 802–811, 2006.

[136] Martin Olsen. Nash stability in additively separable hedonic games and community structures. *Theory of Computing Systems*, 45(4):917–925, 2009.

[137] Martin Olsen. A general view on computing communities. *Mathematical Social Sciences*, 66(3):331–336, 2013.

[138] Morito Oomine, Aleksandar Shurbevski, and Hiroshi Nagamochi. Parameterization of strategy-proof mechanisms in the obnoxious facility game. In *Proceedings of the 10th International Workshop on Algorithms and Computation*, pages 286–297, 2016.

[139] James B. Orlin. Max flows in $O(nm)$ time, or better. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing*, pages 765–774, 2013.

[140] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[141] Christos Papadimitriou. Algorithms, games, and the Internet. In *Proceedings of the 33rd ACM Symposium on Theory of Computing*, pages 749–753, 2001.

[142] David C. Parkes, Ariel D. Procaccia, and Nisarg Shah. Beyond dominant resource fairness: Extensions, limitations, and indivisibilities. *ACM Transactions on Economics and Compution*, 3(1):3:1–3:22, 2015.

[143] Michal Pióro and Deep Medhi. *Routing, Flow, and Capacity Design in Communication and Computer Networks*. Elsevier, 2004.

[144] C. Gregory Plaxton. A simple family of top trading cycles mechanisms for housing markets with indifferences. In *Proceedings of the 24th International Conference on Game Theory*, pages 1–23, 2013.

[145] Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks. *Journal of Graph Algorithms and Applications*, 10(2):191–218, 2006.

[146] Jos A. M. Potters and Stef H. Tijs. The nucleolus of a matrix game and other nucleoli. *Mathematics of Operations Research*, 17(1):164–174, 1992.

[147] Ariel D. Procaccia and Moshe Tennenholtz. Approximate mechanism design without money. In *Proceedings of the 10th ACM Conference on Electronic Commerce*, pages 177–186, 2009.

[148] Alvin Roth. Incentive compatibility in a market with indivisible goods. *Economics letters*, 9(2):127–132, 1982.

[149] Walid Saad, Zhu Han, Mérouane Debbah, Are Hjorungnes, and Tamer Basar. Coalitional game theory for communication networks. *IEEE Signal Processing Magazine*, 26(5):77–97, 2009.

[150] Daniela Saban and Jay Sethuraman. House allocation with indifferences: A generalization and a unified view. In *Proceedings of the 14th ACM Conference on Electronic Commerce*, pages 803–820, 2013.

[151] Abdallah Saffidine and Anaëlle Wilczynski. Constrained swap dynamics over a social network in distributed resource reallocation. In *Proceedings of the 11th International Symposium on Algorithmic Game Theory*, pages 213–225, 2018.

[152] Tuomas Sandholm. Contract types for satisficing task allocation. In *Proceedings of the AAAI Spring Symposium: Satisficing Models*, pages 23–25, 1998.

[153] Paolo Serafino and Carmine Ventre. Heterogeneous facility location without money. *Theoretical Computer Science*, 636:27–46, 2016.

[154] Lloyd Shapley and Herbert Scarf. On cores and indivisibility. *Journal of Mathematical Economics*, 1(1):23–37, 1974.

[155] Bismark Singh. Fairness criteria for allocating scarce resources. *Optimization Letters*, pages 1–9, 2020.

[156] Vaibhav B. Sinha. Algorithms and hardness results for resource allocation and facility location problems. *Thesis, University of Texas at Austin*, 2021.

[157] Oskar Skibski, Szymon Matejczyk, Tomasz Michalak, Michael J. Wooldridge, and Makoto Yokoo. k-coalitional cooperative games. In *Proceedings of the 15th International Conference on Autonomous Agents and Multi-Agent Systems*, pages 177–185, 2016.

[158] Liat Sless, Noam Hazon, Sarit Kraus, and Michael J. Wooldridge. Forming k coalitions and facilitating relationships in social networks. *Artificial Intelligence*, 259:217–245, 2018.

[159] Harold S. Stone. Critical load factors in two-processor distributed systems. *IEEE Transactions on Software Engineering*, 4(3):254–258, 1978.

[160] Shanjiang Tang, Bu-Sung Lee, Bingsheng He, and Haikun Liu. Long-term resource fairness: Towards economic fairness on pay-as-you-use computing systems. In *Proceedings of the 28th ACM International Conference on Supercomputing*, pages 251–260, 2014.

[161] Godfried T. Toussaint. Computing largest empty circles with location constraints. *International Journal of Computer and Information Sciences*, 12(5):347–358, 1983.

[162] Abhishek Verma, Luis Pedrosa, Madhukar Korupolu, David Oppenheimer, Eric Tune, and John Wilkes. Large-scale cluster management

at Google with Borg. In *Proceedings of the 10th European Conference on Computer Systems*, pages 18:1–18:17, 2015.

[163] William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of Finance*, 16(1):8–37, 1961.

[164] Marc Vorsatz and Jordi Massó. Weighted approval voting. *Economic Theory*, 36(1):129–146, 2008.

[165] Naftali Waxman, Sarit Kraus, and Noam Hazon. On maximizing egalitarian value in k-coalitional hedonic games. *arXiv:2001.10772*, 2020.

[166] Deshi Ye, Lili Mei, and Yong Zhang. Strategy-proof mechanism for obnoxious facility location on a line. In *Proceedings of the 21st International Conference on Computing and Combinatorics*, pages 45–56, 2015.

[167] Ryo Yoshinaka. Higher-order matching in the linear lambda calculus in the absence of constants is NP-complete. In *Proceedings of the 16th International Conference on Term Rewriting and Applications*, pages 235–249, 2005.

[168] Hongning Yuan, Kai Wang, Ken C. K. Fong, Yong Zhang, and Minming Li. Facility location games with optional preference. In *Proceedings of the 22nd European Conference on Artificial Intelligence*, pages 1520–1527, 2016.

[169] Qiang Zhang and Minming Li. Strategyproof mechanism design for facility location games with weighted agents on a line. *Journal of Combinatorial Optimization*, 28:756–773, 2014.

[170] Shaokun Zou and Minming Li. Facility location games with dual preference. In *Proceedings of the 14th International Conference on Autonomous Agents and MultiAgent Systems*, pages 615–623, 2015.