Maximum Votes Pareto-Efficient Allocations via Swaps on a Social Network

Fu Li
University of Texas at Austin, TX, USA

Xiong Zheng
University of Texas at Austin, TX, USA

Abstract

In recent work, Gourvès, Lesca, and Wilczynski (IJCAI 17) propose a variant of the classic housing markets model in which the matching between agents and objects evolves through Pareto-improving swaps between pairs of agents who are adjacent in a social network. To explore the swap dynamics of their model, they pose several basic questions concerning the set of reachable matchings, and investigate the computational complexity of these questions when the graph structure of the social network is a star, path, or tree, or is unrestricted.

We are interested in 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 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 various graph structures of the social network: path, star, tree, or being unrestricted. 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.

2012 ACM Subject Classification Theory of computation → Algorithmic game theory

Keywords and phrases Housing markets, Distributed process, Algorithms, Complexity

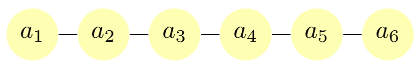
Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.71

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 [22] 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, assuming that any agent is able to trade with any other agent is a quite strong assumption. Agents tend to trade with agents that they know and trust, and it is hard to let two agents who do not trust each other exchange their objects even if they can mutually get benefits. So Gourvès et al. [13] initiated the line of research on the house market problem where the agents are embedded in a social network. 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) via the network. The underlying social network is modeled as an undirected graph. They 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 paper 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 2). In particular, we focus on reachable matchings that are not Pareto-dominated by a reachable matching, rather than by an arbitrary matching as in the standard meaning. 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 endowments.

Maximum Votes Pareto-Efficient Matching. In housing markets, we typically seek a matching between houses and agents that optimizes some social objects. Pareto efficiency has been widely considered in the context of housing allocation [2, 4, 22], which appears to be the minimal requirement for an allocation to be socially acceptable. However, in the context of social network, Pareto efficiency among reachable matchings is not sufficient to be socially acceptable. In the example of Fig. 1, there is a Pareto-efficient matching that improves only two agents, while there is another Pareto-efficient matching that improves everyone, as shown in Fig. 1.



$a_1 : b_2 > b_3 > \boxed{b_1}$ $a_2 : b_1 > b_3 > \boxed{b_2}$

$a_3 : b_4 > b_2 > \boxed{b_3}$ $a_4 : b_5 > b_2 > \boxed{b_4}$

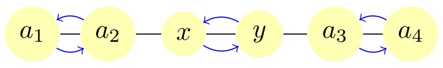
$a_5 : b_6 > b_2 > \boxed{b_5}$ $a_6 : b_2 > \boxed{b_6}$

Figure 1 Consider an instance with six agents in $\{a_1, a_2, a_3, a_4, a_5, a_6\}$ and six objects in $\{b_1, b_2, b_3, b_4, b_5, b_6\}$. The boxed objects represent the initial endowments. Initially, only the swaps between agents a_1 and a_2, and a_2 and a_3 are possible. If we let agents a_1 and a_2 swap their objects first, then we obtain a Pareto-efficient matching in which a_1, a_2 get their best objects. If we let a_2 and a_3 swap their objects first, then additional swaps between pairs (a_1, a_2) and (a_3, a_4) become possible. Indeed, the sequence of swaps, $\{a_2, a_3\}$, $\{a_1, a_2\}$, $\{a_3, a_4\}$, $\{a_4, a_5\}$, and $\{a_5, a_6\}$, yields a different Pareto-efficient matching that improves all agents.

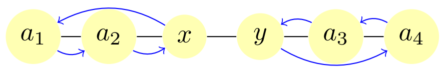
There is a vast literature on refining Pareto-efficiency in object allocation with endowments. Various refining directions are considered, such as fairness ([6, 12]), welfare-maximization ([1, 10]), and stability ([22]). In this work, we propose to refine Pareto-efficiency in the direction of popularity. The notion of popularity was introduced by Abraham et al. [5] for object allocation without endowments. Popularity has gained a lot of attention in the recent past (we refer to the survey by Cseh [11] on this topic). Popular matchings are defined based on comparisons of two matchings with respect to the votes of the agents. Consider an election between two matchings M and M' where vertices 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 a matching is popular if it never loses a head-to-head election against any matching in the voting instance where matchings are candidates and vertices are voters. 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 it prefers the object assigned by this matching 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 to be a reachable matching with the maximum voting number. We remark that maximum votes matchings always exist.

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. 2. Part (a) shows a sequence of swaps that gives a Pareto-efficient matching improving all agents. Part (b) shows a sequence of swaps that also yields a matching that improves all agents, but is not Pareto-efficient. Thus, it is natural to consider the problem of finding a Pareto-efficient matching that has the largest voting number. We refer to this problem as the maximum votes Pareto-efficient (MVPE) matching problem. In particular, the set of MVPE matchings is precisely 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 2 An example in which a non-Pareto-efficient matching improves the most agents. Consider an instance with six agents in $\{a_1, a_2, x, y, a_3, a_4\}$ and six objects in $\{b_1, b_2, b_x, b_y, b_3, b_4\}$. 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.

$a_1 : b_2 > b_x > \boxed{b_1}$	$x : b_y > b_2 > \boxed{b_x}$	$a_3 : b_4 > b_y > \boxed{b_3}$
$a_2 : b_1 > b_x > \boxed{b_2}$	$y : b_x > b_3 > \boxed{b_y}$	$a_4 : b_3 > b_y > \boxed{b_4}$

Related work. Gourvès et al. [13] study these three problems, RO, RM and PE, in four different graph classes, paths, stars, trees, and general graphs. They only consider strict preferences, and thus they consider twelve problems in total. They study these twelve problems with the goal of either exhibiting a polynomial-time algorithm or establishing NP-completeness. 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 problems on stars, for PE on paths, and for RM on trees (which subsumes RM on paths). Gourvès et al. [13] present an elegant reduction from 2P1N-SAT [23] to establish the NP-completeness of RO on trees. They leverage this result to establish the NP-completeness of RM on general graphs via a reduction from RO on trees. Then, they make use of the latter reduction to establish the NP-hardness of PE on general graphs. The work of Gourvès et al. [13] left two of the twelve problems open: RO on paths and PE on trees. Subsequently, two sets of authors independently presented polynomial-time algorithms for RO on paths [14, 7]. To the best of our knowledge, PE on trees remains open.

Huang and Xiao [15] study RO and PE with weak preferences. They establish NP-hardness for RO on paths with weak preferences and give a polynomial-time algorithm for RO on stars with weak preferences. Then, they establish NP-hardness for PE on paths with weak preferences via a reduction from RO on paths with weak preferences. They left PE on stars with weak preferences open. Furthermore, they show that finding a reachable matching maximizing total social welfare is NP-hard for a star or a path with weak preferences.

Bentert et al. [7] consider the case where the preference lists have bounded length. Saffidine and Wilczynski [21] propose an alternative version of RO where an agent can be guaranteed a certain level of satisfaction regardless of the actual exchanges. Müller and Bentert [20] study RM on cliques and cycles. Aspects related to social connectivity are also addressed in recent work on envy-free allocations [8, 9] and on a trade-off between efficiency and fairness [16].

There are several works on ensuring popularity. For settings where a popular matching does not exist, Kavitha et al. [18] 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; and the popularity property is straightforward to carry over; Kavitha et al. [17] show that a popular mixed matching always exists and propose an efficient algorithm to find one. McCutchen [19] 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 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 decomposed into a set of agent intervals, where each agent interval is a set of consecutive agents in the path and all agents in this interval can improve their allocation by swapping objects within the interval. Moreover, given a disjoint union of such agent intervals, we can recover a sequence of swaps improving agents in this disjoint union. Thus, we reduce the problem of computing the maximum votes number to the problem of finding the disjoint union of such agent intervals that covers the most agents; 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 [3] to ensure that each time a dictator agent a is assigned to an object b, where the current partial assignment remains consistent with a maximum votes matching. In particular, we present a subroutine to find the most favorite object b that can be assigned to agent a such that there is a sequence of swaps letting a get b and improving the most agents. In Section 3.4, we use a delicate induction argument to prove the correctness of this subroutine.

In Section 4, we present a simple polynomial-time algorithm for finding a maximum votes Pareto-efficient matching on stars with strict preferences shown

Table 1 Our results and related work. Results in parentheses are implied by other table entries.

	Strict	Weak
Path	poly-time [Section 3]	NP-hard ([15])
Star	poly-time [Section 4]	NP-hard [Section 4]
Tree	NP-hard [Section 5]	(NP-hard)
General	(NP-hard)	(NP-hard)

In terms of negative results, in Section 4, we show that the MVPE problem in star networks with weak preferences is NP-hard by designing a novel reduction from the 3-SAT problem. In Section 5, we prove that the MVPE problem in tree networks with strict preferences is NP-hard by reducing from the RO problem in tree networks with strict preferences, which is known to be NP-complete [13].

Due to space restrictions, some of the proofs are omitted, or are merely sketched. Complete proofs will be provided in the full version of this paper.

2 Preliminaries

We define an object allocation framework (OAF) on a social network as a triple $F = (A, B, \geq, E)$ where A is a set of agents, B is a set of objects such that $|A| = |B|$, $\geq$ is a collection of linear orderings $\{\geq_a\}_{a \in A}$ over B such that $\geq_a$ 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).

We define a matching μ of a given OAF $F = (A, B, \geq, E)$ 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 $F = (A, B, \geq, E)$, any perfect matching μ of F, and any edge $e = (a, a')$ in E, we define an exchange operation on edge e as $\mu' = \mu \setminus \{(a, \mu(a)), (a', \mu(a'))\} \cup \{(a, \mu(a')), (a', \mu(a))\}$, where μ' is the new matching obtained by applying the exchange operation on edge e. We say the exchange is rational if $\mu(a') \geq_a \mu(a)$ and $\mu(a) \geq_{a'} \mu(a')$, 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 sequence of matchings $\mu = \mu_0, \ldots, \mu_k = \mu'$ 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 or star or tree configuration if the social network in F is a path or 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 $\mu(a) \geq_a \mu'(a)$ for all agents a in A, and there is at least one agent b in B such that $\mu(b) >_b \mu'(b)$. A matching is Pareto-efficient if it is not Pareto-dominated by another matching. Throughout this paper, 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:

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

For a given configuration χ, we use $\text{mv}(\chi)$ to denote the largest voting number of a matching in $\text{reach}(\chi)$. We refer to $\text{mv}(\chi)$ as the voting number of χ.

3 Path Network

In this section, we study the maximum votes Pareto-efficient matching problem in a path configuration. We begin by introducing some notations. For any nonnegative integer n, we define $[n]$ as $\{1, \ldots, n\}$. Without loss of generality, in this subsection we restrict attention to OAFs of the form $F = ([n], [n], \geq, \{(b, b+1) \mid 1 \leq b < n\})$ for some positive integer n. We use $[i, j]$ to denote the set of agents between agent i and agent j (inclusive). Let $\Pi = \mu_0, \mu_1, ..., \mu_k$ denote a sequence of matchings 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 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 sequence of swaps 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 2.** *Let* $\Pi = \mu_0, \mu_1, ..., \mu_k$ *denote a sequence of matchings. 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 a canonical sequence of exchanges that assigns object j to agent k by directly moving it along the dipath from j to k. Formally, if $j < k$, $\kappa(j, k) = (j, j+1), (j+1, j+2), ..., (k-1, k)$. If $j \geq k$, $\kappa(j, k) = (j, j-1), (j-1, j-2), ..., (k+1, k)$. A sequence of exchanges Π is said to be a sequence of swaps if all its exchanges are rational, denoted as $\text{rational}(\Pi)$. We remark that $\kappa(j, k)$ was defined, and some associated properties were proved, by Gourvès et. al [13]. The following lemma presents a useful structural property.

▶ **Lemma 3.** *Let* $\chi = (F, \mu)$ *denote a path configuration, let* a *denote a leaf agent in* χ*, and let* a' *denote an agent in* χ*. If there is a matching* μ'' *in* $\text{reach}(\chi)$ *such that* $\mu''(a) = \mu(a')$*, then (1)* $\text{rational}(\kappa(a', a))$ *holds, and (2)* μ'' *belongs to* $\text{reach}((F, \mu'))$*, where* μ' *is the matching reached from* μ *via* $\kappa(a', a)$.

3.1 Maximum Votes Pareto-Efficient Matching

In this section, we present a polynomial-time algorithm for finding a maximum votes Pareto-efficient matching given a path configuration. In order to find an MVPE matching, we first present an algorithm for computing the maximum voting 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 [3]. In the serial dictatorship algorithm, we let agents pick whatever best object they are left with in an arbitrary sequential order. In our case, agents pick objects according to the order of their IDs, i.e., the smallest agent picks first and the largest agent picks last. Assuming that agent a picks object b, we need to meet three additional requirements. First, object b can be assigned to agent a via a sequence of swaps. Second, after object b is assigned to agent a via a sequence of swaps, we obtain a new matching μ' and a new configuration $\chi' = (F, \mu')$. We need to maintain the invariant that there exists a matching τ in $\text{reach}(\chi')$ such that $\text{votes}_F(\tau, \mu) = \text{mv}(\chi)$. Third, object b is the most preferred object of agent a subjects to the first two constraints.

The above three constraints along with the property of serial dictatorship are sufficient for us to prove that the eventual matching we obtain is an MVPE matching. Now, let us first present our algorithm for computing the maximum votes matching of $\text{reach}(\chi)$.

3.2 Maximum Number of Votes

Throughout this section, let $\chi = (F, \mu)$ denote a path configuration. Below we present Algorithm 1, which computes the largest voting number of matching in $\text{reach}(\chi)$.

▶ **Definition 4** (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 R indicates that it moves to the right. Symbol L indicates that it moves to the left. Symbol S indicates that it does not move.*

For any matching ν in $\text{reach}(\chi)$, we define an associated directional sequence as follows. For agent i, if its initial object $\mu(i)$ moves to the left in ν (i.e., the position $\nu^{-1}(\mu)$ of object μ_i is smaller than the initial position i in μ), then the ith symbol in the directional sequence is L, i.e., the object is left-moving. If it is moved to the right, then the ith symbol is R, i.e., the object is right-moving. 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 5** (RL-block). *We define an RL-block as a directional sequence of the form $R^m L^n$ = RRR...RLLL....L where m, n are positive.*

Remark: 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. Therefore, given a matching ν in $\text{reach}(\chi)$, we can decompose $[n]$ into a set of maximal intervals such that all agents in each interval get moved or improved. Let $\text{Decompose}(\nu)$ denote the set of maximal intervals decomposed from $[n]$ according to ν.

▶ **Observation 6.** *Given a matching ν in $\text{reach}(\chi)$, $DS(\nu)$ can be uniquely decomposed into a disjoint union of RL-blocks and S symbols.*

Given a matching ν in $\text{reach}(\chi)$, by applying Observation 6 on the directional sequence $DS(\nu)$, we observe that for any interval $[i, j]$ in $\text{Decompose}(\nu)$, the interval $[i, j]$ can be uniquely partitioned as a set of sub-intervals such that for each sub-interval $[i', j']$, $DS(\nu)[i', j']$ is an RL-block. We refer to this partition as the RL-decomposition of interval $[i, j]$.

Given a maximal interval $[i, j]$ in $\text{Decompose}(\nu)$, let $\text{RLD}([i, j])$ denote the set $\{[i', j'] \subseteq [i, j] \mid DS(\nu)[i', j'] \text{ is an RL-block}\}$ that contains those subintervals of $[i, j]$ that are also RL-blocks. Let $\text{RLD}(\nu)$ denote the set $\bigcup_{[i,j]\in\text{Decompose}(\nu)} \text{RLD}([i, j])$.

▶ **Definition 7** (RL-interval). *We say that an interval $[i, j]$ is an RL-interval if at least one of the following condition holds: (1)* $\text{rational}(\kappa(j, i))$*; (2)* $\text{rational}(\kappa(i, j))$*; (3) there exists an integer k such that $i < k < j$,* $\text{rational}(\kappa(k+1, i))$*, and* $\text{rational}(\kappa(k, j))$*.*

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

▶ **Observation 8.** *For any RL-interval $[i, j]$, there exists a sequence of swaps between agents in $[i, j]$ that improves all agents in the interval.*

▶ **Lemma 9.** *Let ν be a matching in* $\text{reach}(\chi)$ *and let $[i, j]$ be in* $\text{RLD}(\nu)$*, 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. Then, b_1 is the rightmost right-moving object and b_2 is the leftmost left-moving object. Then, from Observations 1 and 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. Thus, we have a reachable matching ν mapping b_1 to leaf agent j, and mapping b_2 to leaf agent i. By Lemma 3, $\kappa(i + m - 1, j)$ and $\kappa(i + m, i)$ are both rational. ◀

Let $\mathcal{I}$ denote the set of all RL-intervals in χ. Let $\Psi(\mathcal{I})$ denote the set $\{X \subseteq \mathcal{I} \mid$ all intervals in X are disjoint$\}$.

▶ **Lemma 10.** *For any matching ν in* $\text{reach}(\chi)$*, the set of intervals* $\text{RLD}(\nu)$ *is an element in $\Psi(\mathcal{I})$.*

Proof. Since $\mathcal{I}$ contains all RL-intervals in χ and each interval in $\text{RLD}(\nu)$ is an RL-interval (Lemma 9), the set of intervals $\text{RLD}(\nu)$ is a subset of $\mathcal{I}$. Also, all intervals in $\text{RLD}(\nu)$ are clearly disjoint. ◀

▶ **Lemma 11.** *For any set of intervals D in $\Psi(\mathcal{I})$, there exists a matching ν in* $\text{reach}(\chi)$ *such that* $\text{RLD}(\nu) = D$*.*

Proof. We explicitly construct such a matching as follows. For each RL-interval $[i, j]$ in D, from Observation 8, we know there exists a sequence of swaps that improves all agents in $[i, j]$. Thus, matching ν can be constructed by applying such a sequence of swaps for each interval $[i, j]$ in D on the initial matching μ. ◀

Let $\text{MCDI}(\mathcal{I})$ denote the maximum coverage of some disjoint intervals in $\mathcal{I}$, i.e., the maximum size of the union of disjoint intervals in $\mathcal{I}$.

▶ **Lemma 12.** $\text{mv}(\chi) = \mathit{MCDI}(\mathcal{I})$*.*

Proof. Lemma 10 implies that $\text{mv}(\chi) \leq \text{MCDI}(\mathcal{I})$. Assume by contradiction that $\text{mv}(\chi) < \text{MCDI}(\mathcal{I})$. Then, there exists a set D in $\Psi(\mathcal{I})$ such that $\text{mv}(\chi) < \sum_{[i,j]\in D} j - i + 1$. By Lemma 11, there exists a matching ν in $\text{reach}(\chi)$ such that $\text{RLD}(\nu) = D$. For matching ν, we have $\text{votes}_F(\nu, \mu) = \sum_{[i,j]\in \text{RLD}(\nu)} j - i + 1 = \sum_{[i,j]\in D} j - i + 1 > \text{mv}(\chi)$, a contradiction. ◀

Using the above lemma, to compute the maximum votes of a configuration, we only need to compute the maximum coverage of disjoint intervals in $\mathcal{I}$. Algorithm 1 has two steps. The nested for loops compute all the RL-intervals $\mathcal{I}$ for a given χ. The second step invokes the maximum disjoint interval algorithm on $\mathcal{I}$.

Algorithm 1 MV(χ).

```
Input: A configuration $\chi = (F, \mu_0)$
Output: The maximum votes number of $\chi$
$\mathcal{I} \leftarrow \emptyset$
for $i \leftarrow 1$ to $n-1$ do
    for $j \leftarrow i+1$ 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 MCDI($\mathcal{I}$)
```

3.3 Weighted Version of Maximum Votes Matching

We remark that there is a polynomial-time algorithm solving a weighted version of finding a maximum votes matching. 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 votes$_F(\mu', \mu, W) = \sum_{a \in [n]: \mu'(a) >_a \mu(a)} W(a)$, i.e., the total weight of the agents that prefer μ' to μ. In addition, let 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 compute the maximal sum of weights of 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 a 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 most weighted path in an acyclic graph.

3.4 MVPE algorithm

In this section, we present Algorithm 3 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 implies that rational$(\kappa(\mu^{-1}(b), a))$ holds. We then apply $\kappa(\mu^{-1}(b), a)$ in χ to let agent a match object b. Hence, we get a new configuration χ' on agents $[a+1, n]$ 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 on the right side of $\mu^{-1}(b)$ in χ', rather than all agents.

Formally, we introduce some 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 mv$(\chi, [i,j])$ denote the maximum number of agents in $[i,j]$

that can be improved by any reachable matching in reach(χ), votes$_F(\mu', \mu, [i,j])$ denote the number of agents in $[i,j]$ that prefer μ' to μ, and MVM($\chi, [i,j]$) denote the set of reachable matchings achieving mv($\chi, [i,j]$).

We are prepared to introduce our recursive subroutine called BOLA (best object for the leftmost agent), shown as Algorithm 2. Given a configuration $\chi = (F, \mu)$ on agents $[a,n]$ and an agent $a' \geq a$, Algorithm 2 finds the leftmost agent a's most preferred object b^* such that there exists a μ' in MVM($\chi, [a',n]$) with $\mu'(a) = b^*$. In Algorithm 2, agent a' is used to limit the objects that agent a can be possibly matched with. Specifically, Algorithm 2 only considers objects that are initially owned by agents in $[a',n]$. In Algorithm 2, we need to compute 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 1 as discussed in Section 3.3.

Fig. 4 shows a possible execution of Algorithm 2. Here, Algorithm 2 runs with a configuration χ on the agents in $[a,n]$ and an agent a' in $[a,n]$, and it finds an object b^* initially owned by agent i that satisfies the two **if** conditions.

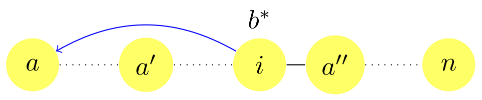


Figure 4 A demo example for Algorithm 2.

Algorithm 2 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 $\mu'$ in MVM($\chi, [a',n]$) assigning $b^*$ to $a$
// Remark: $a$ is the leftmost agent in $\chi$
$b^*, V \leftarrow \mu(a), \mathrm{mv}(\chi, [a',n])$
for $i \leftarrow a$ to $n$ do
    if $\mu(i) \geq_a b^* \wedge \mathrm{rational}(\kappa(i,a))$ then
        // Remark: $i$ denotes an agent
        $\mu^* \leftarrow$ the matching that results from applying $\kappa(i,a)$ to $\mu$
        $\chi^* \leftarrow (F, \mu^*)$
        $a'' \leftarrow \max(i+1, a')$
        if $\mathrm{votes}(\mu^*, \mu, [a', a''-1]) + \mathrm{mv}(\chi^*, [a'', n]) = V$ then
            $b^* \leftarrow \mu(i)$
        end
    end
end
return $b^*$
```

Lemma 13 characterizes all possible maximum votes matchings in terms of the two **if** conditions used in Algorithm 2. Next we use Lemma 13 to prove Lemma 14, which is our main correctness lemma. It implies that to find the leftmost agent a's most preferred object b^* such that $\exists \mu' \in$ MVM($\chi, [a',n]$) : $\mu'(a) = b^*$, we can iterate through all objects that satisfy the two **if** conditions and find the best one for agent a.

▶ **Lemma 13.** *Let $\chi = (F, \mu)$ denote an configuration, let $[a,n]$ denote the set of agents in χ, and let a' denote an agent in $[a,n]$. Let τ denote a matching in* reach(χ). *Let $b = \tau(a)$, and $a'' = \max(\mu^{-1}(b) + 1, a')$. 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$(\mu', \mu, [a', a''-1]) + \text{mv}(\chi'[a+1, n], [a'', n]) = \text{mv}(\chi, [a', n])$, *where* μ' *is obtained from applying* $\kappa(\mu^{-1}(b), a)$ *to* μ *and* $\chi' = (F, \mu')$.

▶ **Lemma 14.** *Algorithm 2 with an input* χ *on agents* $[a, n]$ *and an agent* a' *in* $[a, n]$ *returns the most preferred object* b^* *of agent* a*'s such that there exists a matching* μ' *in MVM*$(\chi, [a', n])$ *with* $\mu'(a) = b^*$.

Proof. It straightforward to verify that the returned b^* satisfies two conditions (1) and (2) in Lemma 13. Therefore, Lemma 13 implies that there exists a matching μ' in MVM$(\chi, [a', n])$ with $\mu'(a) = b^*$.

Then we prove that b^* is the agent a's most preferred object. Assume by contradiction that there is another b is more most preferred to agent a than b^* and $\exists \mu' \in \text{MVM}(\chi, [a', n]) : \mu'(a) = b$. By Lemma 13, two conditions (1) and (2) in Lemma 13 holds and thus the algorithm will return b instead, a contradiction. ◀

We now present Algorithm 3 that uses Algorithm 2 as a building block to find an MVPE matching. In Algorithm 3, we maintain a partial matching τ_i, which is a matching involving only agents in $[1, i]$. At the ith iteration, we invoke the BOLA subroutine to find the agent i's most preferred object b^* that satisfies the condition in Lemma 14. Then we apply $\kappa(b^*, i)$ to assign object b^* to agent i and remove agent i from the configuration afterwards.

Let $P(i)$ denote the predicate: there exists ν in MVM$(\chi, [1, n])$ such that τ_i is a subset of ν. Algorithm 3 maintains the invariant: $P(i)$ holds for all i in $[n]$. This invariant ensures that the final matching τ_n returned by Algorithm 3 is a matching in MVM$(\chi, [1, n])$. The serial dictatorship mechanism ensures that τ_n is also a Pareto-efficient matching.

Algorithm 3 MVPEM(χ).

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

▶ **Theorem 15.** *The matching* τ_n *returned by Algorithm 3 is an MVPE matching.*

4 Star Network

In this section, we consider the case when 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 a center agent and a non-center agent to reduce the MVPE problem to the problem of finding a longest path in a directed acyclic graph.

▶ **Theorem 16.** *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 always involves the center agent. Moreover, notice that any leaf agent can change its object at most once in any valid sequence of swaps. To see this, observe that once an object moves from the center to a leaf, it can never return to the center. Thus, any sequence of swaps can be represented as an ordered list of leaf agents. Let $L = (i_1, i_2, \ldots, i_k)$ denote such an list, where k belongs to $[n]$. Note that $\mu(i_j) <_O \mu(i_{j+1})$ 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.

To find a maximum votes matching in star networks, it is equivalent to find a longest ordered list of leaves corresponding to a sequence of swaps, since a leaf agent gets improved if and only if it is included in the list. This problem reduces to the search of a longest path in a directed acyclic graph $G = (\{O\} \cup [n], E')$. The edge set E' contains a directed edge from a to a' if and only if agent a' belongs to $[n]$, $\mu(a') >_O \mu(a)$, and $\mu(a) >_{a'} \mu(a')$. There is a path from O to a' in the digraph if and only if the center can get the initial endowment of agent a'. It is straightforward to see that there exists a polynomial-time algorithm solving this longest path problem in G.

Next we prove that any maximum votes matching ν is Pareto-efficient. Assume for the sake of contradiction that reachable matching ν' Pareto-dominates ν. Consider the ordered lists of agents L and L' corresponding to the sequences of swaps for reaching ν and ν', respectively. Since ν' Pareto-dominates ν, any agent in L that gets a better allocation in ν also gets a better allocation in ν', i.e., belongs to L'. Furthermore, since ν is a maximum votes matching, the ordered list L is the longest ordered list of agents. Therefore, the list L' is contained in L. (Otherwise, L' contains all agents in L and is longer than L, which contradicts the assumption that L is the longest ordered list of agents.) That is, L' has the same agents as in L, and hence $L' = L$ as all agents in the ordered lists are sorted by the preferences of center agent O. Therefore, we have $\nu' = \nu$, a contradiction. ◀

4.1 Weak Preference

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 sequences of matchings $\Pi = \mu_0, \ldots, \mu_k$, we define the corresponding center object sequence π as the list of objects owned by the center agent O, i.e., $\pi = [\mu_0(O), \ldots, \mu_k(O)]$. We use $\pi[i]$ to denote $\mu_i(O)$, and for any object b, we use $\pi(b)$ to denote the index of first occurrence of b in π, and $\pi(b) = -1$ if b does not belong to π. We have the following observations for the center object sequence.

▶ **Observation 17.** *Let X be a non-center agent with initial endowment x such that x belongs to π. Then, $\pi[\pi(x)-1] \geq_X x$.*

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

▶ **Observation 18.** *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 establish NP-hardness by proving that it is NP-hard to find a maximum votes matching.

Construction. We use a reduction from the 3-SAT. In an instance of 3-SAT, we are given a propositional formula ϕ that is the conjunction of m clauses $C_1, \dots, C_m$. 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 $x_1, \dots, x_n$. 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 largest voting number $2n + 3m$.

We begin by creating the set of agents A. For each variable index i, there are three agents in A: S_i, T_i, F_i. For each clause $C_j = l_{j,0} \cup l_{j,1} \cup l_{j,2}$, there are three agents $P_{j,0}, P_{j,1}, P_{j,2}$ 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 lower case 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, representing whether the corresponding variable x_i is true or false. Furthermore, for each literal $l_{j,k}$ of variable x_w, we associate instance an object $a_{j,k}$ defined as follows. If $l_{j,k} = x_w$, then $a_{j,k} = t_w$; otherwise, $a_{j,k} = f_w$. For example, if $l_{j,k} = x_{15}$, then $a_{j,k} = t_{15}$, and if $l_{j,k} = \bar{x}_7$, then $a_{j,k} = f_7$.

For agent preferences, we only consider the objects that each agent prefers at least as its initial object; other objects can be put behind its initial endowment in any order. The center agent O is indifferent to all objects. We use boxed objects to indicate the initial endowments, and objects in a bracket are indifferent to an agent. For agents S_i, T_i, F_i, their preferences from most preferred to least preferred are $S_i : (t_i, f_i), (o, \boxed{s_i})$, $T_i : s_i, \boxed{t_i}$, $F_i : s_i, \boxed{f_i}$. For each k in $\{0, 1, 2\}$, $P_{j,k} : p_{j,k-1}, a_{j,k}, \boxed{p_{j,k}}$, where $p_{j,-1} = p_{j,2}$.

We now present some properties of our gadgets.

▶ **Lemma 19.** *Let S_i, T_i, F_i be the three agents in the variable gadget corresponding to a variable x_i, $\Pi = \mu_0, \dots, \mu_k$ be a sequence of matchings, and 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. If exactly two agents of $\{S_i, T_i, F_i\}$ are improved, then exactly one of $\{t_i, f_i\}$ belongs to π.*

Proof. Notice that if agent T_i or F_i is improved, then either T_i or F_i gets object s_i by the preferences of T_i and F_i. However, in a matching, s_i can be matched to one agent. Therefore, Π cannot improve 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 of $\{S_i, T_i, F_i\}$ get improved, then exactly one agent in $\{T_i, F_i\}$ is improved, and hence exactly one of $\{t_i, f_i\}$ belongs to π by Observation 18. ◀

▶ **Lemma 20.** *Let $C_j = l_{j,0} \cup l_{j,1} \cup l_{j,2}$ denote a clause. For each k in $\{0, 1, 2\}$, let $P_{j,k}$ denote the agent corresponding to literal $l_{j,k}$, 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 $l_{j,k}$. Let $\Pi = \mu_0, \dots, \mu_k$ be a sequence of matchings, and π denote the corresponding center object sequence of Π. If none of $a_{j,0}, a_{j,1}, a_{j,2}$ is in π, then none of $p_{j,0}, p_{j,1}, p_{j,2}$ is π.*

Proof. We prove the contrapositive. Assume that there is 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 $\pi(p_{j,k})$ where k belongs to $\{0, 1, 2\}$. By Observation 17, $\pi[\pi(p_{j,k}) - 1] \geq_{P_{j,k}} p_{j,k}$. 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 $p_{j,k}$ is the object with minimum $\pi(p_{j,k})$, $\pi[\pi(p_{j,k}) - 1]$ 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 $a_{j,k}$. ◀

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

Proof. By Lemma 19, there are at least n agents that are not improved. Moreover, the center agent O initially holds one of most preferred objects, and hence cannot be improved. Therefore, there are at most $2n + 3m$ agents that are improved. If χ_ϕ achieves the largest voting number $2n + 3m$, then there are exactly two agents that are improved in each variable gadget and all three agents that are improved in each clause gadget.

By Lemma 19, for all i in $[n]$, there is exactly one object in $\{t_i, f_i\}$ in π. If t_i is in π, then let variable x_i be true, otherwise let x_i be false. Therefore, we get an assignment A_π from π. We then show that A_π is a satisfiable assignment, i.e., each clause C_j in ϕ is true with respect to assignment A_π. For the sake of contradiction, assume that there exists a clause C_j that is false. Hence, all of the literals $l_{j,0}, l_{j,1}, l_{j,2}$ in C_j are false under assignment A_ϕ. Let $a_{j,0}, a_{j,1}, a_{j,2} \in \{t_i, f_i \mid i \in [n]\}$ be the objects associated with literals $l_{j,0}, l_{j,1}, l_{j,2}$, respectively. We deduce that none of $a_{j,0}, a_{j,1}, a_{j,2}$ is in π. By Lemma 20, π does not contain $\{p_{j,0}, p_{j,1}, p_{j,2}\}$. That is, agents $P_{j,0}, P_{j,1}, P_{j,2}$ is not improved by π due to Observation 18. ◀

▶ **Lemma 22.** *If there is a satisfiable assignment for* ϕ*, then* $\mathrm{mv}(\chi_\phi) = 2n + 3m$*.*

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

Now we show how to obtain a sequence of swaps according to objects $(b_1, \ldots, b_n)$. Let $\mathcal{C}$ denote a temporary set, which is initialized to be the set of all clauses $\{C_1, C_2, \ldots, C_m\}$. 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. Perform two swaps $(O, S_i), (O, B_i)$ to let center agent O get b_i and S_i get object o.
2. If there are a clause C_j in $\mathcal{C}$ and a literal $l_{j,k}$ in C_j such that object b_i is the associated object $a_{j,k}$, then we preform the following steps:
   - Perform these swaps in sequence: $(O, P_{j,k}), (O, P_{j,k\oplus 1}), (O, P_{j,k\oplus 2}), (O, P_{j,k})$, where $\oplus$ denotes the addition module 3. Remark that by doing so, agent $P_{j,k}$ gets its most preferred object for all k in $\{0, 1, 2\}$ and center agent O still holds b_i.
   - Remove the clause C_j from $\mathcal{C}$.
3. In the end, perform the swap (O, S_i). Remark: After this swap, agent O gets object o and S_i get one of its most preferred object 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. First of all, by 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. Then, we consider agents in clause gadgets. Since A_ϕ is a satisfiable assignment for ϕ, for all clause C_j there exists some literal $l_{j,k}$ of variable x_i such that $l_{j,k}$ is true under assignment A_ϕ. Therefore, by the definition of object $a_{j,k}$, we deduce that $a_{j,k}$ is b_i. Thus, all agents associated with clause C_j are improved. That is, the above sequence of swaps improves all agents in clause gadgets. There are $3m$ agents in clause gadgets. To sum up, there are $2n + 3m$ agents that are improved in total. ◀

▶ **Theorem 23.** *Finding an MVPE matching on stars with weak preferences is NP-hard.*

5 Tree Network

▶ **Theorem 24.** *Finding an MVPE matching on trees with strict preferences is NP-hard.*

We establish Theorem 24 by proving that it is NP-hard to find a maximum votes matching. The latter result is obtained via a reduction to reachable object problem on trees, which was shown to be NP-complete by Gourvès et al. [13].

6 Conclusion

To refine Pareto-efficiency, we study the complexity of finding a maximum votes Pareto-efficient matching when a group of agents exchange their objects along a social network. By presenting two polynomial-time algorithms and two NP-hardness results, we shed light on the frontiers between tractable and intractable cases in terms of the structures of the social network and whether ties in preference are allowed. A challenging open question is to completely characterize the social network structures for which case the MVPE problem is polynomial-time solvable.

References

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 and Tayfun Sönmez. Random serial dictatorship and the core from random endowments in house allocation problems. *Econometrica*, 66(3):689–701, 1998.

3 Atila Abdulkadiroğlu and Tayfun Sönmez. House allocation with existing tenants. *Journal of Economic Theory*, 88(2):233–260, 1999.

4 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.

5 David J. Abraham, Robert W. Irving, Telikepalli Kavitha, and Kurt Mehlhorn. Popular matchings. *SIAM Journal on Computing*, 37(4):1030–1045, 2007.

6 Haris Aziz, Serge Gaspers, Simon Mackenzie, and Toby Walsh. Fair assignment of indivisible objects under ordinal preferences. *Artificial Intelligence*, 227:71–92, 2015.

7 Matthias Bentert, Jiehua Chen, Vincent Froese, and Gerhard J. Woeginger. Good things come to those who swap objects on paths. *arXiv*, 2019. `arXiv:1905.04219`.

8 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.

9 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.

10 Péter Biró and Jens Gudmundsson. Complexity of finding Pareto-efficient allocations of highest welfare. *European Journal of Operational Research*, 291(2):614–628, 2021.

11 Ágnes Cseh. Trends in computational social choice. In U. Endriss, editor, *Trends in Computational Social Choice*, chapter 6, pages 105–122. lulu.com, 2017.

12 Federico Echenique, Antonio Miralles, and Jun Zhang. Fairness and efficiency for probabilistic allocations with endowments. *arXiv*, 2019. `arXiv:1908.04336`.

13 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.

14 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.

15 Sen Huang and Mingyu Xiao. Object reachability via swaps under strict and weak preferences. *Autonomous Agents and Multiagent Systems*, 34(2):51, 2020.

16 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.

17 Telikepalli Kavitha, Julián Mestre, and Meghana Nasre. Popular mixed matchings. *Theoretical Computer Science*, 412(24):2679–2690, 2011.

18 Telikepalli Kavitha, Meghana Nasre, and Prajakta Nimbhorkar. Popularity at minimum cost. *Journal of Combinatorial Optimization*, 27(3):574–596, 2014.

19 Richard Matthew McCutchen. The least-unpopularity-factor and least-unpopularity-margin criteria for matching problems with one-sided preferences. In *Proceedings of 8th Latin American Symposium on Theoretical Informatics*, pages 593–604, 2008.

20 Luis Müller and Matthias Bentert. On reachable assignments in cycles and cliques. *arXiv*, 2020. `arXiv:2005.02218`.

21 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.

22 Lloyd Shapley and Herbert Scarf. On cores and indivisibility. *Journal of Mathematical Economics*, 1(1):23–37, 1974.

23 Ryo Yoshinaka. Higher-order matching in the linear lambda calculus in the absence of constants is NP-complete. In Jürgen Giesl, editor, *Proceedings of 16th International Conference on Term Rewriting and Applications*, pages 235–249, 2005.