Size versus truthfulness in the
House Allocation problem[*]

Piotr Krysta[1], David Manlove[2], Baharak Rastegari[3] and Jinshan Zhang[1]

[1] *Department of Computer Science, University of Liverpool,*
Ashton Building, Ashton Street, Liverpool L69 3BX, UK.
Email: `p.krysta@liverpool.ac.uk.`

[2] *School of Computing Science, University of Glasgow,*
Sir Alwyn Williams Building, Glasgow G12 8QQ, Glasgow, UK.
Email: `david.manlove@glasgow.ac.uk.`

[3] *School of Electronics and Computer Science, University of Southampton*
Email: `b.rastegari@soton.ac.uk.`

Abstract

We study the House Allocation problem (also known as the Assignment problem), i.e., the problem of allocating a set of objects among a set of agents, where each agent has ordinal preferences (possibly involving ties) over a subset of the objects. We focus on truthful mechanisms without monetary transfers for finding large Pareto optimal matchings. It is straightforward to show that no deterministic truthful mechanism can approximate a maximum cardinality Pareto optimal matching with ratio better than 2. We thus consider randomised mechanisms. We give a natural and explicit extension of the classical Random Serial Dictatorship Mechanism (RSDM) specifically for the House Allocation problem where preference lists can include ties. We thus obtain a universally truthful randomised mechanism for finding a Pareto optimal matching and show that it achieves an approximation ratio of $\frac{e}{e-1}$. The same bound holds even when agents have priorities (weights) and our goal is to find a maximum weight (as opposed to maximum cardinality) Pareto optimal matching. On the other hand we give a lower bound of $\frac{18}{13}$ on the approximation ratio of any universally truthful Pareto optimal mechanism in settings with strict preferences. In the case that the mechanism must additionally be non-bossy with an additional technical assumption, we show by utilizing a result of Bade that an improved lower bound of $\frac{e}{e-1}$ holds. This lower bound is tight since RSDM for strict preference lists is non-bossy. We moreover interpret our problem in terms of the classical secretary problem and prove that our mechanism provides the best randomised strategy of the administrator who interviews the applicants.

Keywords: House allocation problem; Assignment problem; Pareto optimal matching; Randomised mechanisms; Truthfulness

1   Introduction

We study the problem of allocating a set of indivisible objects among a set of agents. Each agent has private ordinal preferences over a subset of objects — those they find acceptable, and each agent may be allocated at most one object. This problem has been studied by both economists and computer scientists. When monetary transfers are not permitted, the problem is referred to as the *House Allocation problem* (henceforth abbreviated by HA [20, 1, 38] or the *Assignment problem* [19, 9] in the literature. In this paper we opt for the term House Allocation problem. Most of the work in the literature assumes that the agents have strict preferences over their acceptable objects. However, it often happens though that an agent is indifferent between two or more objects. Here we let agents express indifference, and hence preferences may involve ties unless explicitly stated otherwise.

It is often desired that as many objects as possible become allocated among the agents — i.e., an allocation of maximum size is picked, hence making as many agents happy as possible. Usually, depending on the application of the problem, we are required to consider some other optimality criteria, sometimes instead of, and sometimes in addition to, maximising the size of the allocation. Several optimality criteria have been considered in the HA setting, and perhaps the most studied such concept is *Pareto optimality* (see, e.g., [1, 2, 15, 12, 33]), sometimes referred to as Pareto efficiency. Economists, in particular, regard Pareto optimality as the most fundamental requirement for any "reasonable" solution to a non-cooperative game. Roughly speaking, an allocation μ is Pareto optimal if there does not exist another allocation μ' in which no agent is worse off, and at least one agent is better off, in μ'. In this work we are mainly concerned with Pareto optimal allocations of maximum size, but will also consider weighted generalisations.

The related *Housing Market* problem (HM) [29, 30, 34] is the variant of HA in which there is an *initial endowment*, i.e., each agent owns a unique object initially (in this case the numbers of agents and objects are usually defined to be equal). In this setting, the most widely studied solution concept is that of the *core*, which is an allocation of agents to objects satisfying the property that no coalition C of agents can improve (i.e., every agent in C is better off) by exchanging their own resources (i.e., the objects they brought to the market). In the case of strict preferences, the core is always non-empty [30], unique, and indeed Pareto optimal. When preferences may include ties, the notion of *core* that we defined is sometimes referred to as the *weak core*. In this case a core allocation need not be Pareto optimal. Jaramillo and Manjunath [21], Plaxton [27], and Saban and Sethuraman [32] provide polynomial-time algorithms for finding a core allocation that does additionally satisfy Pareto optimality. Our problem differs from HM in that there is no initial endowment, and hence our focus is on Pareto optimal matchings rather than outcomes in the core.

For strictly-ordered preference lists, Abraham et al. [2] gave a characterisation of Pareto optimal matchings that led to an $O(m)$ algorithm for checking an arbitrary matching for Pareto optimality, where m is the total length of the agents' preference lists. This characterisation was extended to the case that preference lists may include ties by Manlove [24, Sec. 6.2.2.1], also leading to an $O(m)$ algorithm for checking a matching for Pareto optimality. For strictly-ordered lists, a maximum cardinality Pareto optimal matching can be found in $O(\sqrt{n_1}m)$ time, where n_1 is the number of agents [2]. The fastest algorithm currently known for this problem when preference lists may include ties is based on minimum cost maximum cardinality matching and has complexity $O(\sqrt{n}m \log n)$ (see, e.g., [24, Sec. 6.2.2.1], where n is the total number of agents and objects.

As stated earlier, agents' preferences are private knowledge. Hence, unless they find it in their own best interests, agents may not reveal their preferences truthfully. An allocation mechanism is *truthful* if it gives agents no incentive to misrepresent their preferences.

2

Perhaps unsurprisingly, a mechanism based on constructing a maximum cardinality Pareto optimal allocation is manipulable by agents misrepresenting their preferences (Theorem 2.3 in Section 2). Hence, we need to make a compromise and weaken at least one of these requirements. In this work, we relax our quest for finding a maximum cardinality Pareto optimal allocation by trading off the size of a Pareto optimal allocation against truthfulness; more specifically, we seek truthful Pareto optimal mechanisms that provide good approximations to the maximum size.

Under strict preferences, Pareto optimal matchings can be computed by a classical algorithm called the *Serial Dictatorship Mechanism* (SDM) (see, e.g., [1]), also referred to as the *Priority Mechanism* (see, e.g., [9]). SDM is a straightforward greedy algorithm that takes each agent in turn and allocates to him the most preferred available object on his preference list. Precisely due to this greedy approach, SDM is truthful. Furthermore, SDM is guaranteed to find a Pareto optimal allocation that has size at least half that of a maximum one, merely because *any* Pareto optimal allocation has size at least half that of a maximum one (see, e.g., [2]). Hence, at least in the case of strict preferences, we are guaranteed an approximation ratio of 2. Can we do better? It turns out that if we stay in the realm of deterministic mechanisms, a 2-approximation is the best we can hope for (Theorem 2.3, Section 2).

Hence we turn to randomised mechanisms in order to achieve a better approximation ratio. The obvious candidate to consider is the Random Serial Dictatorship Mechanism (RSDM) (see, e.g., [1]), also known as the Random Priority mechanism (see, e.g., [9]), that is defined for HA instances with strict preferences. RSDM randomly generates an ordering of the agents and then proceeds by running SDM relative to this ordering.

When indifference is allowed, finding a Pareto optimal allocation is not as straightforward as for strict preferences. For example, one may consider breaking the ties randomly and then applying SDM. This approach, unfortunately, may produce an allocation that is not Pareto optimal. To see this, consider a setting with two agents, 1 and 2, and two objects, o_1 and o_2. Assume that 1 finds both objects acceptable and is indifferent between them, and that 2 finds only o_1 acceptable. The only Pareto optimal matching for this setting is the one in which 1 and 2 are assigned o_2 and o_1 respectively. Assume that 1 is served first and that, as both objects are equally acceptable to him, is assigned o_1 (after an arbitrary tie-breaking). Therefore when 2's turn arrives, there is no object remaining that he finds acceptable, and is hence left unmatched, resulting in a matching that is not Pareto optimal.

Few works in the literature have considered extensions of SDM to the case where agents' preferences may include ties. However Bogomolnaia and Moulin [10] and Svensson [35] provide an implicit extension of SDM (in the former case for *dichotomous preferences*[1]) but do not give an explicit description of an algorithm. Aziz et al. [4] provide an explicit extension for a more general class of problems, including HA. Pareto optimal matchings in HA can also be found by reducing to the HM setting [21], which involves creating dummy objects as endowments for the agents. This allows one of the aforementioned algorithms for HM [21, 27, 32] to be utilised to find a Pareto optimal matching in the core. However the reduction increases the instance size, and in particular the number of agents n_1 and the maximum length of a tie in any agent's preference list. Consequently, even the fastest truthful Pareto optimal mechanism for HM, that of Saban and Sethuraman [32], has time complexity no better than $O(n_1^3)$ in the worst case.

Contributions of this paper. In this paper we provide a natural and explicit extension of SDM for the setting in which preferences may exhibit indifference. We argue that our

---

[1]An agent's preference list is *dichotomous* if it comprises a single tie containing all acceptable objects.

extension is more intuitive than that of Aziz et al. [4] when considering specifically HA. Moreover, as the mechanism of Saban and Sethuraman [32] does not consider the agents sequentially, it is difficult to analyse its approximation guarantee. Our algorithm runs in time $O(n_1^2\gamma+m)$, where γ is the maximum length of a tie in any agent's preference list and m is the total length of the agents' preference lists. This is faster than the algorithm in [32] when $\gamma = o(n_1)$ and $m = o(n_1^3)$. We prove the following results that involve upper and lower bounds for the approximation ratio (relative to the size of a maximum cardinality Pareto optimal matching) of randomised truthful mechanisms for computing a Pareto optimal matching:

1. By extending RSDM to the case of preference lists with ties, we give a universally truthful randomised mechanism[2] for finding a Pareto optimal matching that has an approximation ratio of $\frac{e}{e-1}$ with respect to the size of a maximum cardinality Pareto optimal matching.

2. We give a lower bound of $\frac{18}{13}$ on the approximation ratio of any universally truthful Pareto optimal mechanism in settings with strict preferences. If the mechanism must additionally be non-bossy, with an additional technical assumption[3], we observe that [7] implies an improved lower bound of $\frac{e}{e-1}$. This lower bound is tight since the classical RSDM mechanism for strict preferences is non-bossy.

3. We extend RSDM to the setting where agents have priorities (weights) and our goal is to find a maximum weight (as opposed to maximum cardinality) Pareto optimal matching. Our mechanism is universally truthful and guarantees a $\frac{e}{e-1}$-approximation with respect to the weight of a maximum weight Pareto optimal matching.

4. We finally observe that our problem has an "online" or sequential flavour similar to secretary problems[4]. Given this interpretation, we prove that our mechanism uses the best random strategy of interviewing the applicants in the sense that any other strategy would lead to an approximation ratio worse than $\frac{e}{e-1}$ (see also below under related work).

Discussion of technical contributions. As observed above via a simple example, SDM with arbitrary tie breaking need not lead to a Pareto optimal matching in general. Indeed, the presence of indifference in agents' preference lists introduces major technical difficulties. This is because decisions with respect to objects in one tie cannot be committed to when an agent is considered, as they may block some choices for future agents. When extending SDM from strict preferences to preferences with ties, we first present an intuitive mechanism, called SDMT-1, based on the idea of augmenting paths. It is relatively easy to prove that SDMT-1 is Pareto optimal and truthful. We also show that SDMT-1 is

---

[2]A randomised mechanism is universally truthful if it is a probability distribution over truthful deterministic mechanisms. This is the strongest known notion of truthfulness for randomised mechanisms.

[3]A deterministic mechanism in settings with strict preferences is non-bossy if no agent can misreport his preferences in such a way that his allocation is not changed but the allocation of some other agent is changed. This additional assumption restricts the class to randomised mechanisms that are symmetrizations of truthful, Pareto optimal and non-bossy mechanisms, see Section 6 for details.

[4]In the secretary problem, an administrator is willing to hire the best secretary out of n rankable applicants for a position. The applicants are interviewed one-by-one in random order. A decision about each particular applicant is to be made immediately after the interview. Once rejected, an applicant cannot be recalled. During the interview, the administrator can rank the applicant among all applicants interviewed so far, but is unaware of the quality of yet unseen applicants. The question is about the optimal strategy to maximise the probability of selecting the best applicant.

able to generate any given Pareto optimal matching. However, it is difficult to analyse the approximation guarantee of the randomised version of SDMT-1. For this purpose we build on the primal-dual analysis of Devanur et al. [16]. They employ a linear programming (LP) relaxation of the bipartite weighted matching problem. They prove that their dual solution is feasible in expectation for the dual LP and use it to show the approximation guarantee. Towards this goal they prove two technical lemmas, a dominance lemma and a monotonicity lemma. The randomised version of SDMT-1 uses random variables Y_i for each agent $i \in N$ to generate a random order in which agents are considered. Considering agent i and fixed values of the random variables Y_{-i} of all other agents, Devanur et al. [16] define a threshold y^c, which as Y_i varies determines when agent i is matched (to an object) or unmatched (dominance lemma). (Note that we will denote the threshold y^c as θ.) The monotonicity lemma shows how values of the dual LP variables change when Y_i varies. To extend the definition of y^c, we need to remember the structure of all potential augmenting paths in SDMT-1, and for this purpose we introduce a second mechanism, SDMT-2. Interestingly, SDMT-2 is inspired by the idea of top trading cycle mechanisms, see, e.g., [32], however it retains the "sequential" nature of SDMT-1. The two mechanisms, SDMT-1 and SDMT-2, are equivalent: they match the same agents, giving them objects from the same ties. This implies that SDMT-2 is also truthful and Pareto optimal. SDMT-2 is the key to defining the threshold y^c: its running time is no worse than that of SDMT-1, but it implicitly maintains all relevant augmenting paths arising from agents' ties. We prove the monotonicity and dominance lemmas for SDMT-2 by carefully analysing the structure of frozen subgraphs that are generated from the relevant augmenting paths; here frozen roughly means that they will not change subsequently. Finally, we would like to highlight that our proof of an $\frac{18}{13}$ lower bound on the approximation ratio of any universally truthful and Pareto optimal mechanism uses Yao's minmax principle and an interesting case analysis to account for all such possible mechanisms.

Related work. This work can be placed in the context of designing truthful approximate mechanisms for problems in the absence of monetary transfer [28]. Bogomolnaia and Moulin [9] designed a randomised weakly truthful and envy-free mechanism, called the Probabilistic Serial mechanism (PS), for HA with complete lists. Very recently the same authors considered the same approximation problem as ours but in the context of envy-free rather than truthful mechanisms, and for strict preference lists and unweighted agents [11]. They showed that PS has an approximation ratio of $\frac{e}{e-1}$, which is tight for any envy-free mechanism. Bhalgat et al. [8] investigated the social welfare of PS and RSDM. Tight deterministic truthful mechanisms for weighted matching markets were proposed by Dughmi and Ghosh [18] and they also presented an $O(\log n)$-approximate random truthful mechanism for the Generalised Assignment Problem (GAP) by reducing, with logarithmic loss in the approximation, to the solution for the value-invariant GAP. In subsequent work Che et al. [14] provided an $O(1)$-approximation mechanism for GAP. Aziz et al. [5] studied notions of fairness involving the stochastic dominance relation in the context of HA, and presented various complexity results for problems involving checking whether a fair assignment exists. Chakrabarty and Swamy [13] proposed *rank approximation* as a measure of the quality of an outcome and introduced the concept of *lex-truthfulness* as a notion of truthfulness for randomised mechanisms in HA.

RSDM is related to online bipartite matching algorithms. The connection was observed by Bhalgat et al. [8], who noted the similarity between RSDM and the RANKING algorithm of Karp et al. [22]. Karp et al. [22] proved that the expected size of the matching given by their RANKING algorithm is at least $\frac{e-1}{e}$ times the optimal size. Bhalgat et al. [8] observed that RSDM will essentially behave the same way as RANKING for instances

of HA where the agents' preference lists relate to the order in which the objects arrive. Hence for this family of instances an approximation ratio of $\frac{e}{e-1}$ holds for RSDM.

The weighted version of our problem is related to two widely-studied online settings, known in the literature as the online vertex-weighted bipartite matching problem [3] and secretary problems [6]. In our problem the administrator holds all the objects (they can be thought of as available positions), and all agents with unknown preference lists are applicants for these objects. Each applicant also has a private weight, which can be thought of as their quality (reflecting the fact that some of an agent's skills may not be evident from their CV, for example). However we assume that they cannot overstate their weights (skills), because they might be checked and punished. This is similar to the classical assumption of no overbidding (e.g., in sponsored search auctions). Applicants are interviewed one-by-one in a random order. When an applicant arrives he chooses his most-preferred available object and the decision as to whether it is allocated to him is made immediately, and cannot be changed in the future.

Our weighted agents correspond to weighted vertices in the vertex-weighted bipartite matching context, but our objects do not arrive online as in the setting of [3]. However, if the preference ordering of each agent in our setting, over his acceptable objects, coincides with the arrival order of the objects in [3], then the two problems are the same. In the transversal matroid secretary problem, see, e.g., [17], objects are known in advance as in our setting, weighted agents arrive in a (uniform) random order, and the goal is to match them to previously unmatched objects. The administrator's goal is to find a (random) arrival order of agents that maximises the ratio between the total weight of matched agents and the maximum weight of a matching if all the agents preference lists are known in advance. We show that even if the weights of all agents are the same, our algorithm uses the best possible random strategy; no other such strategy leads to better than $\frac{e}{e-1}$-approximate matching.

Organisation of the paper. The remainder of the paper is organised as follows. In Section 2 we define notation and terminology used in this paper, and show the straightforward lower bound for the approximation ratio of deterministic truthful mechanisms. SDMT-1 and SDMT-2 are presented in Sections 3 and 4 respectively, and in the latter section it is proved that the two mechanisms are essentially equivalent. The approximation ratio of $\frac{e}{e-1}$ for the randomised version of the two mechanisms is established in Section 5, whilst Section 6 contains our lower bound results. Finally, some concluding remarks are given in Section 7.

2 Definitions and preliminary observations

Let $N = \{1, 2, \cdots, n_1\}$ be a set of n_1 *agents* and $O = \{o_1, o_2, \ldots, o_{n_2}\}$ be a set of n_2 *objects*. Let $n = n_1 + n_2$. Let $[i]$ denote the set $\{1, 2, \cdots, i\}$. We assume that each agent $i \in N$ finds a subset of objects acceptable and has a preference ordering, not necessarily strict, over these objects. We write $o_t \succ_i o_s$ to denote that agent i *strictly prefers* object o_t to object o_s, and write $o_t \simeq_i o_s$ to denote that i is *indifferent between* o_t and o_s. We use $o_t \succeq_i o_s$ to denote that agent i either strictly prefers o_t to o_s or is indifferent between them, and say that i *weakly prefers* o_t to o_s. In some cases a weight w_i is associated with each agent i, representing the priority or importance of the agent. Weights need not be distinct. Let $W = (w_1, w_2, \ldots, w_{n_1})$. To simplify definitions, we assume that all agents are assigned weight equal to 1 if we are in an unweighted setting.

We assume that the indifference relation is transitive. This implies that each agent essentially divides his acceptable objects into different bins or *indifference classes* such

that he is indifferent between the objects in the same indifference class and has a strict preference ordering over these indifference classes. For each agent i, let C_k^i, $1 \leq k \leq n_2$, denote the kth indifference class, or *tie*, of agent i. We also assume that if there exists $l \in [n_2]$, where $C_l^i = \emptyset$, then $C_q^i = \emptyset$, $\forall q, l \leq q \leq n_2$. We let $L(i) = (C_1^i \succ_i C_2^i \succ_i \cdots \succ_i C_{n_2}^i)$ and call $L(i)$ the *preference list* of agent i. We abuse notation and write $o \in L(i)$ if o appears in preference list $L(i)$, i.e., if agent i finds object o acceptable. We say that agent i ranks object o in kth position if $o \in C_k^i$. We denote by $rank(i, o)$ the rank of object o in agent i's preference list and let $rank(i, o) = n_2 + 1$ if o is not acceptable to i. Therefore $o_t \succ_i o_s$ if and only if $rank(i, o_t) < rank(i, o_s)$, and $o_t \simeq_i o_s$ if and only if $rank(i, o_t) = rank(i, o_s)$.

Let $L = (L(1), L(2), \cdots, L(n_1))$ denote the joint preference list profile of all agents. We write $L(-i)$ to denote the joint preference list profile of all agents except agent i; i.e., $L(-i) = (L(1), \ldots, L(i-1), L(i+1), \ldots, L(n_1))$. Let $\mathcal{L}$ denote the set of all possible joint preference list profiles. An instance of HA is denoted by $I = (N, O, L, W)$. We drop W and write $I = (N, O, L)$ if we are dealing with an instance where agents are not assigned weights, or equivalently if they all have the same weight. Let $\mathcal{I}$ denote the set of all possible instances of HA.

A *matching* μ is a subset of $N \times A$ such that each agent and object appears in at most one pair of μ. If $(i, o) \in \mu$, agent i and object o are said to be *matched together*, and o is the *partner* of i and vice versa. If $(i, o) \in \mu$ for some o, we say that i is *matched*, and *unmatched* otherwise. The definitions of *matched* and *unmatched* for an object are analogous. If agent i is matched, $\mu(i)$ denotes the object matched to i. Similarly if object o is matched, $\mu^{-1}(o)$ denotes the agent matched to o. In what follows, we will refer to the *underlying graph* of I, which is the undirected graph $G_0 = (V, E)$ where $V = N \cup O$ and $E = \{(i, o), i \in N, o \in L(i)\}$. We also use μ to denote a matching (in the standard graph-theoretic sense) in G_0. The *size* of a matching μ is equal to the number of agents matched under μ. In the presence of weights, the *weight* of a matching is equal to the sum of the weights of the matched agents.

For two given matchings μ_1, μ_2, we will frequently use $\mu_1 \oplus \mu_2$ to denote the symmetric difference with respect to their sets of edges. An *alternating path* in G_0, given a matching μ_1, is a path that consists of edges that alternately belong to μ_1 and do not belong to μ_1. An *augmenting path* in G_0 is an alternating path where the first and the last vertices on the path are unmatched in μ_1. To *augment* along an augmenting path, given matching μ_1, means that a new matching μ_2 is created by removing edges on the path that belong to μ_1 and adding edges on the path that do not belong to μ_1.

A matching μ is *Pareto optimal* if there is no other matching under which some agent is better off while none is worse off. Formally, μ is *Pareto optimal* if there is no other matching μ' such that (i) $\mu'(i) \succeq_i \mu(i)$ for all $i \in N$, and (ii) $\mu'(i') \succ_{i'} \mu(i')$ for some agent $i' \in N$. Manlove [24, Sec. 6.2.2.1] gave a characterisation of Pareto optimal matchings in instances of HA (potentially with ties) in terms of a number of graph-theoretic structures, which we will now define.

An *alternating path coalition* w.r.t. μ comprises a sequence $P = \langle i_0, i_1, \ldots, i_{r-1}, o_k \rangle$, for some $r \geq 1$, where i_j is a matched agent ($0 \leq j \leq r - 1$) and o_k is an unmatched object. If $r = 1$ then i_0 strictly prefers o_k to $\mu(i_0)$. Otherwise, if $r \geq 2$, i_0 strictly prefers $\mu(i_1)$ to $\mu(i_0)$, i_j weakly prefers $\mu(i_{j+1})$ to $\mu(i_j)$ ($1 \leq j \leq r - 2$), and i_{r-1} weakly prefers o_k to $\mu(i_{r-1})$.

An *augmenting path coalition* w.r.t. μ comprises a sequence $P = \langle i_0, i_1, \ldots, i_{r-1}, o_k \rangle$, for some $r \geq 1$, where i_0 is an unmatched agent and o_k is an unmatched object. If $r = 1$ then i_0 finds o_k acceptable. Otherwise, if $r \geq 2$, i_j is a matched agent ($1 \leq j \leq r - 1$), i_0 finds $\mu(i_1)$ acceptable, i_j weakly prefers $\mu(i_{j+1})$ to $\mu(i_j)$ ($1 \leq j \leq r - 2$), and i_{r-1} weakly

prefers o_k to $\mu(i_{r-1})$.

A *cyclic coalition* w.r.t. μ comprises a sequence of applicants $P = \langle i_0, i_1, \ldots, i_{r-1} \rangle$, for some $r \geq 2$, all matched in μ, such that i_j weakly prefers $\mu(i_{j+1})$ to $\mu(i_j)$ for each j ($0 \leq j \leq r-1$), and i_j strictly prefers $\mu(i_{j+1})$ to $\mu(i_j)$ for some j ($0 \leq j \leq r-1$) (all subscripts are taken modulo r when reasoning about cyclic coalitions).

Proposition 2.1 ([24]). *Given an instance I of HA and a matching μ in I, μ is Pareto optimal if and only if μ admits no* alternating path coalition, *no* augmenting path coalition, *and no* cyclic coalition.

Let $\mathcal{M}$ denote the set of all possible matchings. A *deterministic mechanism* ϕ maps an instance of HA to a matching, i.e., $\phi : \mathcal{I} \to \mathcal{M}$. Let $R : \mathcal{M} \to [0, 1]$ denote a distribution over possible matchings (which we also call a *random matching*); i.e., $\sum_{\mu \in \mathcal{M}} R(\mu) = 1$. A *randomised mechanism* ϕ is a mapping from $\mathcal{I}$ to a distribution over possible matchings, i.e., $\phi : \mathcal{I} \to Rand(\mathcal{M})$, where $Rand(\mathcal{M})$ is the set of all random matchings. A deterministic mechanism is *Pareto optimal* if it always returns a Pareto optimal matching. A randomised mechanism is *Pareto optimal* if it always returns a distribution over Pareto optimal matchings.

Agents' preferences are private knowledge and an agent may prefer not to reveal his preferences truthfully if it is not in his best interests, for a given mechanism. A deterministic mechanism ϕ is *dominant strategy truthful* (or *truthful*) if agents always find it in their best interests to declare their preferences truthfully, no matter what other agents declare, i.e., for every joint preference list profile L, for every agent i, and for every possible declared preference list $L'(i)$ for i, $\phi(L(i), L(-i)) \succeq_i \phi(L'(i), L(-i))$. A randomised mechanism ϕ is *universally truthful* if it is a probability distribution over deterministic truthful mechanisms.

Denote by $w(\phi(I))$ the (expected) weight of the (random) matching generated by mechanism ϕ on instance $I \in \mathcal{I}$, and by $w(I)$ the weight of a maximum weight matching in I. The *approximation ratio* of ϕ is then defined as $\max_{I \in \mathcal{I}} \frac{w(I)}{w(\phi(I))}$. Note that a maximum weight matching has the same weight as a maximum weight Pareto optimal matching, as the following proposition shows.

Proposition 2.2. *Given an instance I of HA, a maximum weight matching has the same weight as a maximum weight Pareto optimal matching.*

Proof. We provide a procedure for transforming a maximum weight matching μ in I to a Pareto optimal matching μ' in I with the same weight.

Let G_0 be the underlying graph for I and let G_0' be the subgraph of G_0 induced by $N' \cup A$, where N' is the set of agents who are matched in μ. Define the cost of each edge (i, o_j) in G_0' to be $rank(i, o_j)$. Find a maximum cardinality matching μ' of minimum cost in G_0'. It is easy to see that μ and μ' are of the same cardinality and have the same weight, as they each match all agents in N'. It remains to show that μ' is Pareto optimal in I.

If μ' is not Pareto optimal in I then by Proposition 2.1, μ' admits a coalition C that is either an alternating path coalition, or an augmenting path coalition, or a cyclic coalition. If C is an augmenting path coalition then $\mu' \oplus C$ has larger weight than μ, a contradiction as μ is a maximum weight matching. Hence C is an alternating path coalition or a cyclic coalition. In either case let $\mu'' = \mu' \oplus C$. Then $|\mu''| = |\mu'|$ but the cost of μ'' is less than the cost of μ', a contradiction as μ' is a maximum cardinality minimum cost matching in G_0'. Hence μ' is Pareto optimal in I. $\square$

We now give a straightforward lower bound for the approximation ratio of any deterministic truthful mechanism for HA with strict preferences.

Theorem 2.3. *No deterministic truthful mechanism for HA can achieve approximation ratio better than* 2. *The result holds even for strict preferences.*

Proof. Consider an HA instance I with two agents, 1 and 2, and two objects, o_1 and o_2. Assume that both agents have weight 1 and strictly prefer o_1 to o_2. Then I admits two matchings of size (weight) 2. Assume, for a contradiction, that there exists a truthful mechanism ϕ with approximation ratio strictly smaller than 2. Then in I, ϕ must pick one of the two matchings of size 2. Assume, without loss of generality, that ϕ picks $\mu = \{(1, o_2), (2, o_1)\}$. Now, assume that agent 1 misrepresents his acceptable objects and declares o_1 as the only object acceptable to him. Let I' denote the instance of HA so obtained. As ϕ is truthful, when executed on I' it must not assign o_1 to 1, or else 1 finds it in his best interests to misrepresent his preferences as he would strictly prefer his allocated object in I' to his allocated object in I. Hence ϕ must return a matching of size at most 1 (by assigning an object to agent 2) when applied to I'. However, I' admits a matching of size 2, namely $\mu' = \{(1, o_1), (2, o_2)\}$. Therefore the approximation ratio of ϕ is at least 2, a contradiction. $\qquad\square$

Corollary 2.4. *No deterministic truthful Pareto optimal mechanism for HA can achieve approximation ratio better than* 2. *The result holds even for strict preferences.*

As mentioned in Section 1, the upper bound of 2 is achievable via SDM for HA with strict preferences [2]. If weights and ties exist, simply ordering the agents in decreasing order of their weights and running SDMT-1 (see Algorithm 1 in Section 3) or SDMT-2 (see Algorithm 2 in Section 4) gives a deterministic truthful and Pareto optimal mechanism with approximation ratio 2 (Theorem 3.6 in Section 3). This resolves the problem for deterministic mechanisms and motivates looking into relaxing our requirements. In the following sections we look for randomised truthful mechanisms that construct 'large' weight Pareto optimal matchings.

3 First truthful mechanism: SDMT-1

3.1 Introduction

When preferences are strict, SDM produces a Pareto optimal matching. However when indifference is allowed, finding an arbitrary Pareto optimal matching is not as straightforward as in the case of strict preferences, as illustrated via an example in Section 1.

In Section 3.2 we introduce SDMT-1, Serial Dictatorship Mechanism with Ties, a mechanism that generalises SDM to the case where agents' preferences may involve ties. Then in Section 3.3, we show that SDMT-1 is truthful and is guaranteed to produce a Pareto optimal matching. We further show that SDMT-1 is capable of generating any given Pareto optimal matching.

3.2 Mechanism SDMT-1

Let $I = (N, O, L)$ be an instance of HA, and let a fixed order σ of the agents be given. Assume, w.l.o.g., that $\sigma(i) = i$ for all agents $i \in N$. The formal description of SDMT-1 is given in Algorithm 1; an informal description follows.

SDMT-1 constructs an undirected bipartite graph $G = (V, E)$ where $V = N \cup O$ and the set of edges E changes during the execution of SDMT-1; initially $E = \emptyset$. The mechanism returns a matching μ; initially $\mu = \emptyset$. It then proceeds in n_1 phases, where each phase corresponds to one iteration of the for loop in Algorithm 1. In phase i, agent i is considered and the objects in i's preference list are examined in the order of the indifference classes

---
Algorithm 1: Serial Dictatorship Mechanism with Ties, version 1 (SDMT-1)

Input: Agents N; Objects O; Preference list profile L; An order of agents σ

Output: Matching μ

Let $G = (N \cup O, E)$, $E \leftarrow \emptyset$, $\mu \leftarrow \emptyset$.

for *each agent $i \in N$ in the order of σ* **do**

> Let $\ell \leftarrow 1$
>
> Step (*): **if** $C_\ell^i \neq \emptyset$ **then**
>
> > $E \leftarrow E \cup \{(i,o) : o \in C_\ell^i\}$; // all new edges are non-matching edges
> >
> > **if** there is an augmenting path from i in G **then**
> >
> > > augment along this path and update μ accordingly; // i is provisionally allocated some $o \in C_\ell^i$ and (i,o) is now a matching edge
> >
> > **end**
>
> **else**
>
> > $E \leftarrow E \setminus \{(i,o) : o \in C_\ell^i\}$
> >
> > $\ell \leftarrow \ell + 1$; Go to Step (*)
>
> **end**

end

end

Return μ; //each matched agent is allocated his matched object

---

they belong to. Recall that C_ℓ^i denotes the ℓ'th indifference class of agent i. When objects $o \in C_\ell^i$ are examined, edges (i,o) are provisionally added to E for all $o \in C_\ell^i$. We then check whether μ admits an augmenting path in G that starts from agent i. If such a path exists, we augment along that path and modify μ accordingly. This would mean that agent i is assigned some $o \in C_\ell^i$ and every other agent already matched is assigned an object that he ranks in the same indifference class his previous object. Otherwise – if μ admits no augmenting path in G – edges (i,o) are removed from E for all $o \in C_\ell^i$. In general, once an agent i is assigned an object $o \in C_\ell^i$ he will remain matched in μ, although he may be required to exchange o for another object in C_ℓ^i in order to allow a newly-arrived agent to receive o.

Notice that, at any stage of the mechanism, an edge (i,o) belongs to E if and only if *either* agent i is matched in μ and $o \simeq_i \mu(i)$, *or* SDMT-1 is at phase i and examining the indifference class to which o belongs. Therefore, it is fairly straightforward to observe the following.

Observation 3.1. *At the end of phase i of SDMT-1, if agent i is assigned no object then he will be assigned no object when SDMT-1 terminates. Otherwise, if i is provisionally assigned an object o, then he will be allocated an object that he ranks the same as o in the final matching.*

3.3   Properties of SDMT-1

Before proceeding to prove our main claim, namely that SDMT-1 is truthful and produces a Pareto optimal matching, let us discuss a relevant concept that is both interesting in its own right and useful in the proofs that follow. In practice agents may have priorities and the mechanism designer may wish to ensure that the agents with higher priorities are served before satisfying those with lower priorities. Roth et al. [31] studied this concept under the term *priority matchings* in the case where each agent's preference list is one single tie. This work was motivated by the kidney exchange problem in which patients are assigned priorities based on various criteria; e.g., children and hard-to-match patients have

higher priorities. Prior to Roth et al. [31], Svensson [35] studied a similar concept under the name *queue allocation* in a setting similar to ours. We formally define this concept using the terminology *strong priority matching*, reflecting both the definition in [31] and the fact that preference lists are more general than single ties.

In general, assume that we are given an ordering of the agents $\sigma = i_1, \ldots, i_{n_1}$. However, recall that in this section we are assuming, without loss of generality, that $i_j = j$, i.e., $\sigma = 1, 2, \ldots, n_1$. For each matching μ, the *signature* of μ w.r.t. σ, denoted by $\rho(\mu, \sigma)$, is a vector $\langle \rho_1, \ldots, \rho_{n_1} \rangle$ where for each $i \in [n_1]$, $\rho_i = rank(i, \mu(i))$ if i is matched under μ, and $\rho_i = n_2 + 1$ otherwise. A matching μ is a *strong priority matching (SPM)* w.r.t. σ if $\rho(\mu, \sigma)$ is lexicographically minimum, taken over all matchings μ. That is, (i) the highest priority agent 1 has one of his first-choice objects (assuming $L(1) \neq \emptyset$); (ii) subject to (i), there is no matching μ' such that $\mu'(2) \succ_2 \mu(2)$, where 2 is the agent with the second-highest priority; (iii) subject to (i) and (ii), there is no matching μ'' such that $\mu''(3) \succ_3 \mu(3)$, where 3 is the agent with the third-highest priority, etc. It is easy to see that a given HA instance may admit more than one SPM w.r.t. σ, but all of them have the same signature. When σ is fixed and known, we simply say that μ is an SPM.

Theorem 3.2. *The matching produced by SDMT-1 is a strong priority matching w.r.t. σ.*

Proof. Let μ_k denote the matching at the end of phase k (hence $\mu_{n_1} = \mu$). Assume, for a contradiction, that the claim does not hold. Hence μ is not an SPM in I. Let μ^* be an SPM in I. Let i be the first agent in σ (i.e., the lowest-indexed agent) who strictly prefers his partner under μ^* to his partner under μ, i.e., $\mu^*(i) \succ_i \mu(i)$ and $\mu^*(j) \simeq_j \mu(j)$, $\forall j < i$ (we denote this fact by **D1**). Therefore, in phase i of SDMT-1 no augmenting path has been found starting from (i, o), for any object o such that $o \succeq_i \mu^*(i)$ (we denote this fact by **D2**). Also, it follows from **D1** and Observation 3.1 that, $\mu^*(j) \simeq_j \mu_{i-1}(j)$, $\forall j < i$ (we denote this fact by **D3**).

Let G^* denote the graph G in phase i during the examination of the indifference class to which $\mu^*(i)$ belongs. By **D2**, G^* must admit no augmenting path w.r.t. μ_{i-1}. We show, however, that G^* admits an augmenting path starting from i. To see this note that, by **D1** and **D3**, and by the construction of edges E, edges $(j, \mu^*(j))$ belong to G^* $\forall j < i$. If $\mu^*(i)$ is unmatched in μ_{i-1} then $(i, \mu^*(i))$ constitutes an augmenting path of size 1 in G^*. Otherwise, let j_1 denote the partner of $\mu^*(i)$ under μ_{i-1} (note that $j_1 < i$). It follows from **D1** and **D3**, and the construction of E, that j_1 is matched under μ^*. If $\mu^*(j_1)$ is unmatched under μ_{i-1} then we have found an augmenting path of length 3. Otherwise, let j_2 denote the partner of $\mu^*(j_1)$ under μ_{i-1} (note that $j_2 < i$). The same argument we used for j_1 can be used for j_2, resulting in either the discovery of an augmenting path of size 5 or reaching a new agent. We can repeatedly use this argument and each time we either find an augmenting path (and stop) or visit an agent that appears in σ before i. As each agent is assigned at most one object in every matching, and vice versa, the agents j_r that we encounter on our search for an augmenting path are all distinct. Therefore, since there are a finite number of agents and objects, we are bound to reach an object o that is unmatched under μ_{i-1}, hence exposing an augmenting path in G^*, a contradiction. $\square$

Corollary 3.3. *The matching produced by SDMT-1 is a Pareto optimal matching.*

Proof. By Theorem 3.2, SDMT-1 produces an SPM. It follows from Theorems 1 and 2 in [35] that any SPM is a Pareto optimal matching. Hence, the matching produced by SDMT-1 is a Pareto optimal matching. $\square$

SDMT-1 is truthful, no matter which augmenting path is selected in each phase of the mechanism, as the next result shows. The proof idea is as follows. Note that when an

agent's turn arrives, SDMT-1 assigns him an object from what the algorithm identifies as his "best possible indifference class"; i.e., the top-most indifference class from which he can be assigned an object without harming any previously-arrived agent. Then as soon as he is assigned an object, by Observation 3.1, he is guaranteed to be allocated the same object, or one that he equally values, when the algorithm terminates. Hence, as long as we can show that the algorithm correctly identifies these "best possible indifference classes", it is straightforward to see that no agent can benefit from misreporting. The proof of the next theorem formalises this argument.

Theorem 3.4. *The mechanism SDMT-1 is truthful.*

Proof. Assume, for a contradiction, that the claim does not hold. Let i be the first agent in σ (i.e., the lowest-indexed agent) who benefits from misrepresenting his preferences and reporting $L'(i)$ instead of $L(i)$. Let $L' = (L'(i), L(-i))$.

Let μ denote the matching returned by SMDT-1 on instance $I = (N, O, L)$, i.e., the instance in which agent i reports truthfully, and let μ^* denote the matching returned on instance $I' = (N, O, L')$. Then in I, $\mu^*(i) \succ_i \mu(i)$ and $\mu(j) \succeq_j \mu^*(j)$, $\forall j < i$.

By Theorem 3.2, μ is an SPM in I, and μ^* is an SPM in I'. Suppose that in I, $\mu(j) \succ_j \mu^*(j)$, for some $j < i$. Let k be the smallest integer such that $\mu(k) \succ_k \mu^*(k)$ in I. As $k < i$, for each j ($1 \leq j \leq k$), agent j has the same preference list in I and I', by construction of L'. Hence μ^* cannot be an SPM in I' after all, a contradiction.

It follows that in I, $\mu^*(i) \succ_i \mu(i)$ and $\mu(j) \simeq_j \mu^*(j)$, $\forall j < i$. We now obtain a contradiction to the fact that μ is an SPM in I. $\square$

We now show a bound on the time complexity of SDMT-1. Let γ denote the size of the largest indifference class for a given instance I.

Theorem 3.5. *SDMT-1 terminates in time $O(n_1^2 \gamma + m)$.*

Proof. For each agent i matched under μ, let ℓ_i denote the length of the indifference class to which $\mu(i)$ belongs. Let $|L(i)|$ denote the length of agent i's preference list, $\forall i \in N$. Searching for an augmenting path in a graph $G = (V, E)$ can be done in time $O(|E|)$ using Breadth-First Search (BFS). Hence the search for an augmenting path in each phase i can be done in time $O(\ell_1 + \ell_2 + \cdots + \ell_{i-1} + |L(i)|)$. Therefore SDMT-1 terminates in time $O((n_1 - 1) \cdot \ell_1 + (n_1 - 2)\ell_2 + \cdots + \ell_{n_1 - 1} + \sum_{i \in N} |L(i)|)$. However, $\ell_i \leq \gamma$, $\forall i \in N$, therefore $(n_1 - 1) \cdot \ell_1 + (n_1 - 2)\ell_2 + \cdots + \ell_{n_1 - 1} + \sum_{i \in N} |L(i)| \leq n_1^2 \gamma + m$, where m is the number of (agent,object) acceptable pairs. Hence SDMT-1 terminates in time $O(n_1^2 \gamma + m)$. $\square$

As noted in Section 1, in the strict preferences case, any Pareto optimal matching is at least half the size of a maximum size such matching. The same is true in the general case with indifferences, since any Pareto optimal matching is a maximal matching in the underlying bipartite graph G_0 for I, and any maximal matching in G_0 is at least half the size of a maximum matching in G_0 [23]. Hence SDMT-1 obviously achieves approximation ratio 2 when we are concerned with the cardinality of the matching. We next show that, when agents are assigned arbitrary weights, SDMT-1 achieves the same approximation ratio (relative to a maximum weight Pareto optimal matching) if the agents are ordered in σ in non-increasing order of their weights.

Theorem 3.6. *SDMT-1 achieves approximation ratio of 2 relative to the size of a maximum weight Pareto optimal matching, if the agents are ordered in σ in non-increasing order of their weights.*

Proof. Given an HA instance I, let μ be the matching produced by SDMT-1 and let μ' be a maximum weight Pareto optimal matching in I. List the agents matched under each of these matchings in non-increasing order of weight. Let $i_1, \ldots, i_k$ denote such an order under μ, and let $i'_1, \ldots, i'_l$ denote such an order under μ'.

Take any agent i'_r who is matched under μ', to say o, but not matched under μ (if no such agent exists then μ is itself a maximum weight Pareto optimal matching). Note that, as μ is Pareto optimal, o must be matched under μ, for otherwise $\mu \cup \{(i'_r, o)\}$ Pareto dominates μ. As SDMT-1 generates an SPM w.r.t. σ (Theorem 3.2) and agents are listed in non-increasing order of weight under σ, it follows that o must be allocated in μ to an agent i_s who has at least as large a weight as i'_r (for otherwise $(\mu \setminus \{(i_s, o)\}) \cup \{(i'_r, o)\}$ has a lexicographically smaller signature than μ, a contradiction).

We claim that i_s must be matched under μ' as well, as otherwise $(\mu' \setminus \{(i'_r, o)\}) \cup \{(i_s, o)\}$ has a higher weight than μ', a contradiction. (Recall that a maximum weight Pareto optimal matching must be a maximum weight matching as well by Proposition 2.2.) Hence we have established that, for each agent i'_r matched under μ' but not matched under μ, there exists a unique agent i_s, with weight at least as large as that of i'_r, who is matched under μ. Thus if N_1 is the set of agents matched in μ' and N_2 is the set of agents matched in μ, it follows that $wt(N_2) \geq wt(N_1 \setminus N_2)$, where $wt(N')$ is the sum of the weights of the agents in N', for $N' \subseteq N$. Also $wt(N_2) = wt(N_2 \setminus N_1) + wt(N_2 \cap N_1) \geq wt(N_2 \cap N_1) = wt(N_1) - wt(N_1 \setminus N_2) \geq wt(N_1) - wt(N_2)$, hence the result. $\square$

It is known (see, e.g., [2]) that, in the case of strict preferences, not only can we find a Pareto optimal matching using SDM, but we can also generate *all* Pareto optimal matchings by executing SDM on all possible permutations of the agents. In other words, given any Pareto optimal matching μ, there exists an order of the agents such that executing SDM on that order returns μ. A similar characterisation of Pareto optimal matchings holds in the case of preferences with ties. This is stated by the following result, whose proof is given in the Appendix.

Theorem 3.7. *Any Pareto optimal matching can be generated by some execution of SDMT-1.*

4 Randomised mechanism with weights and ties

In this section we will analyse our mechanism for the weighted version of our problem. Our algorithm in the next section is truthful with respect to agents' preferences and weights (under the no-overbidding assumption, see also the beginning of Section 6) and provides an $\frac{e}{e-1}$-approximate Pareto optimal matching. We will show in Section 6 that, even if the weights of all agents are the same our algorithm uses the best possible random strategy – no other such strategy leads to better than $\frac{e}{e-1}$-approximate matching.

4.1 Second truthful mechanism: SDMT-2

The approximation ratio analysis of the randomised version of SDMT-1 is complex, because it requires additional information which is not maintained by SDMT-1. For the sake of the analysis, we introduce a variant of SDMT-1, called SDMT-2. After introducing some terminology we present SDMT-2, and then establish the equivalence between SDMT-1 and SDMT-2. Pareto optimality and truthfulness of SDMT-2 will then follow from this equivalence and these same two properties of SDMT-1. We will prove that the randomised version of SDMT-2 is $\frac{e}{e-1}$-approximate. By the equivalence of the two algorithms, a randomised version of SDMT-1 has the same approximation ratio.
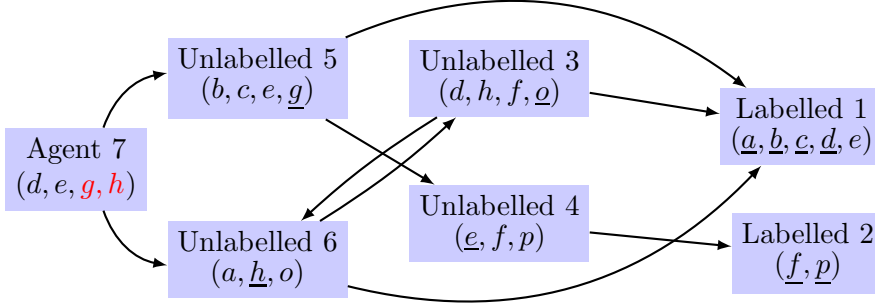
Figure 1: The trading graph $TG(7, \{g, h\}, [6])$, $\underline{h}$ denotes h is owned currently by the agent. Objects in the parentheses below each agent represent a single indifference class of that agent. The common order of the objects is $a \succ b \succ c \succ d \succ e \succ f \succ g \succ h \succ o \succ p$ (used in the text below).

Let $o_1 \succ o_2 \succ \cdots \succ o_{n_2}$ be a common order of all the objects. This order will be used to break possible ties in SDMT-2. In what follows we will use use lower case letters from the beginning of the alphabet to name individual objects, e.g., a, b, c, d, e, f, g, h. We define now some notions that will be used to describe algorithm SDMT-2. These definitions will refer to any time point during an execution of this algorithm. In the course of the algorithm agents will be (temporarily) *allocated* subsets of objects from their preference list. When an agent is allocated a subset of objects we say that he *owns* these objects. Let $S \subseteq N$ and suppose that some of the agents in S have been allocated some objects and the allocated objects to each agent appear in the same indifference class of this agent. At any time during the execution of the algorithm, each agent who is allocated more than one object is called *labelled* and *unlabelled* otherwise. Likewise, at any point during the execution of the algorithm, let $i \in N$, and let $B \subseteq L(i)$ be such that i is not currently allocated any object in B. The *trading graph (TG)* is a directed graph $TG(i, B, S)$ with $\{i\} \cup S$ as the set of nodes, and arcs defined as follows: Let agent i point to each agent in S who owns any object in B. For each unlabelled agent, e.g., $j \in S$, to which i points, suppose the current object allocated to j is in j's kth indifference class C_k^j. Then let j point to each agent in S who currently owns any object in C_k^j not owned by j. Continue this process for the new pointed-to and unlabelled agents until no agent in S needs to point to other agents. See Figure 1 for an example of how $TG(7, \{g, h\}, [6])$ is constructed: agent 7 points to agent 5 and 6 since currently agent 5 owns g and agent 6 owns h; then, as agent 5 is unlabelled, agent 5 points to agents 4 and 1 since agent 4 owns e and agent 1 owns b and c; similarly, agent 6 points to agents 1 and 3; agent 3 points to 1 and 6; only agents 1 and 2 are labelled.

Let $H = \{a \in L(i) \,|\, \text{there is a (directed) path from } i \text{ to a labelled agent in } TG(i, a, S)\}$. Note that, as labelled agents do not point to any agents, no intermediate agent on a directed path is labelled. Note that H may be empty, and it can be found, for instance,[5] by breadth first search (BFS). If $H \neq \emptyset$, let ℓ be the highest indifference class of i with $H \cap C_\ell^i \neq \emptyset$. Define $\max TG(i, L(i), S)$ to be the highest order object in $H \cap C_\ell^i$ (e.g., in Figure 1, $\max TG(7, \{d, e, g, h\}, [6]) = g$). We also explicitly define $\max TG(i, L(i), S) = \emptyset$ if $H = \emptyset$. If $\max TG(i, L(i), S) \neq \emptyset$, then there is a path from i to a labelled agent

---

[5]Here, what only matters is the reachability, that is, existence of such directed path in $TG(i, a, S)$ from agent i to a labelled agent.

Algorithm 2: Serial Dictatorship Mechanism with Ties, version 2 (SDMT-2)

Input: Agents N; Objects O; Preference list profile L; An order of agents σ, w.l.o.g. let $\sigma(i) = i$, $\forall i \in N$

Output: Matching

Let $O_1 \leftarrow O$ // O_1 is the set of currently unallocated objects

for *each agent $i \in N$ in the order of σ* **do**

> Define $j_1 = \begin{cases} \min\{j : O_i \cap C_j^i \neq \emptyset\} & \text{if } O_i \cap L(i) \neq \emptyset; \\ n_2 + 1 & \text{otherwise.} \end{cases}$
>
> $j_2 = \begin{cases} \min\{j : \max TG(i, L(i), [i-1]) \in C_j^i\} & \text{if } \max TG(i, L(i), [i-1]) \neq \emptyset; \\ n_2 + 1 & \text{otherwise.} \end{cases}$
>
> **if** $j_1 \leq j_2$ **then**
>
> > Allocate all the objects in $O_i \cap C_{j_1}^i$ to i; Label i if $|O_i \cap C_{j_1}^i| \geq 2$;
> >
> > $O_{i+1} \leftarrow O_i \backslash (O_i \cap C_{j_1}^i)$
>
> **end**
>
> **else**
>
> > $Trading(i, \max TG(i, L(i), [i-1]), [i-1])$; $O_{i+1} \leftarrow O_i$
>
> **end**

end

For each labelled agent, allocate to him the highest order object he currently owns.

For each unlabelled agent, if he currently owns an object, allocate it to him.

Output the matching.

---

in $TG(i, a, S)$, which can be found by BFS. Suppose the path is $(i_0, i_1, i_2, \cdots, i_k)$, where $i_0 = i$ and only i_k is labelled. Now denote $Trading(i, a, S)$ to be a procedure that allocates the object owned by i_{s+1} to i_s, for $s = 0, 1, \cdots, k-1$. Note that i_k may own more than one object for which i_{k-1} has pointed to i_k. In this case, the highest order object among such objects is allocated to i_{k-1}. After trading, if i_k still owns more than one object, keep i_k labelled and unlabel i_k otherwise. In Figure 1, considering procedure $Trading(7, g, [6])$, we note that there are two paths from agent 7 to a labelled agent: $(7, 5, 1)$ and $(7, 5, 4, 2)$. Procedure $Trading(7, g, [6])$ can use any of those two paths. If $Trading(7, g, [6])$ uses the first path, then it allocates g to agent 7 and b to agent 5, since $b \succ c$, and keeps agent 1 labelled. If procedure $Trading(7, g, [6])$ uses the second path, then it allocates g to agent 7 and e to agent 5, anf f to agent 4, since $f \succ p$, and changes agent 2 to unlabelled.

Recall that $C_{n_2+1}^i = \emptyset$, $\forall i \in [n_1]$. With these preliminaries, we present our algorithm SDMT-2 (see Algorithm 2). In the following, we will refer to kth iteration of the "for loop" in SDMT-2 as the kth loop. Observe that in the kth loop, j_1 is the highest indifference class of i where i can obtain unallocated objects, and j_2 is the highest indifference class of i where i can obtain objects from the allocated objects without hurting the agents prior to i.

Observation 4.1. *For each agent i, after i's turn in "for loop" of SDMT-2, if i is allocated no object, then he will be allocated no object when SDMT-2 terminates. Otherwise, if i is provisionally allocated some objects in his turn, then in the final matching he will be allocated an object in the same indifference class as his initially allocated objects.*

Observation 4.2. *For each agent i, after i's turn, if i is allocated an object $o \in C_j^i$, then all the objects in $\cup_{k=1}^j C_k^i$ have been allocated to either i or to some agents prior to i. Once an object is allocated, it remains allocated until the end of the for loop.*

Now we establish the equivalence of SDMT-1 and SDMT-2.

15

Theorem 4.3. *Given the same input, SDMT-1 and SDMT-2 match the same set of agents. Furthermore, for each matched agent i, the object allocated to i in SDMT-1 is in the same indifference class of i as the object allocated to him in SDMT-2. This equivalence between SDMT-1 and SDMT-2 holds for any fixed common order $\succ$ of the objects used in SDMT-2 and it is also independent of how SDMT-2 finds the directed paths from agent i to a labelled agent in the trading graph $TG(i, L(i), [i-1])$.*

Proof. We will prove the following two facts inductively which simply implies the conclusion of Theorem 4.3. Without loss of generality, suppose the order of agents is $\sigma(i) = i$, $\forall i \in N$. Until the step i,

1. for each agent $k \leq i-1$, the allocated objects of SDMT-1 and SDMT-2 to k are in the same indifference class, (if one of them is empty, the other is empty as well)

2. for each $\ell \leq n_2$, and $a \in C_\ell^i$, there is an augmenting path starting from (i, a) in SDMT-1 if and only if a is unallocated in SDMT-2 or there is a path from i to a labelled agent in $TG(i, a, [i-1])$ in SDMT-2.

Consider the base case, for agent 1, obviously property 1 is true since they are all empty sets. Now, for property 2, let $\ell \leq n_2$ and $a \in C_\ell^1$, then there is an augmenting path from $(1, a)$ in SDMT-1 and a is unallocated in SDMT-2. This shows both implications of property 2 for agent 1.

For the proof of the induction step, suppose properties 1 and 2 are true for all the steps $k \leq i-1$, we now prove that they are true for step i. For property 1, by inductive hypothesis, property 1 holds for any $k \leq i-2$. Since property 2 holds for agent $i-1$ by inductive hypothesis, the objects allocated to agent $i-1$ in SDMT-1 and SDMT-2 will be in the same indifference class, thus, property 1 holds for step i. Now property 2 will be proved true for agent i, for each $\ell \leq n_2$, and $a \in C_\ell^i$:

For $\Rightarrow$ direction, if there is an augmenting path starting from (i, a) in SDMT-1, and if a is allocated previously in SDMT-2, suppose the new matching generated in SDMT-1 due to the augmenting path is $(k, \mu(k))$, $k \leq i$, where $\mu(i) = a$. By property 1 of inductive hypothesis and Observation 4.2, all the objects in $\{\mu(k), k \leq i\}$ have been allocated to some agents $k \leq i-1$ in SDMT-2. For object b, we use $\nu^{-1}(b)$ to denote the agent whom b is allocated to in SDMT-2. Now consider the following path in $TG(i, a, [i-1])$: let $i_1 = \nu^{-1}(\mu(i))$, and if i_1 is labelled then we are done, otherwise, let $i_2 = \nu^{-1}(\mu(i_1))$. If i_2 is labelled, then we are done, otherwise continue this process. Finally, we will reach by this process a labelled agent among the agents in $[i-1]$. This is true because of the pigeonhole principle: i objects from $\{\mu(k), k \leq i\}$ are allocated in SDMT-2 to $i-1$ agents in $[i-1]$.

For $\Leftarrow$ direction, now suppose a is unallocated or there is a path from i to a labelled agent in $TG(i, a, [i-1])$ in SDMT-2. Suppose a is allocated and there is a path from i to a labelled agent in $TG(i, a, [i-1])$. Then by $Trading(i, a, [i-1])$, we can make all the agents $k \leq i$ allocated at least one object and i is allocated a. This defines an allocation of (sets of) objects to agents $k \leq i$ in SDMT-2. Let us now select any matching using this allocation, e.g., $M = \{(k, \nu(k)), k \leq i\}$, where $\nu(i) = a$ (we can also select such a matching if a is unallocated in SDMT-2). For instance, matching ν can assign the highest order object to each agent $k \leq i-1$ from the current set of objects allocated to k, and assign object a to agent i. Suppose the matching generated after step $i-1$ in SDMT-1 is $M' = \{(k, \mu(k)), k \leq i-1\}$. By property 1 of inductive hypothesis, we know $\mu(k)$ and $\nu(k)$ are in the same indifference class of agent k, for any $k \in [i-1]$. Now consider $M \oplus M'$, which consists of alternating paths and cycles. Then a connected component of $M \oplus M'$ that contains $(i, \nu(i))$ must be an odd length alternating path in $M \oplus M'$ w.r.t. M',

---
Algorithm 3: Random SDMT-2 for Weighted Agents with Ties

  Input: Agents N; Objects O; Preference list profile L; Weights W
  Output: Matching
  for *each agent* $i \in N$ **do**
  | Pick $Y_i \in [0,1]$ uniformly at random;
  end
  Sort agents in decreasing order of $w_i(1 - e^{Y_i-1})$ (break ties in favour of smaller index);
  Run SDMT-2 according to above order;
  Return the matching;
---

implying an augmenting path starting from (i,a) in SDMT-1. The argument showing that the connected component that contains $(i, \nu(i))$ must be an odd length alternating path is as follows. If $\nu(i) = a$ is unallocated in SDMT-1, then $(i, \nu(i))$ is an odd length alternating path. Otherwise, suppose $i_1 = \mu^{-1}(\nu(i))$, then consider whether $\nu(i_1)$ is allocated or not in SDMT-1. If not we get an odd path $(i, \nu(i), i_1, \nu(i_1))$. Otherwise continue the search, and let $i_2 = \mu^{-1}(\nu(i_1))$, then consider whether $\nu(i_2)$ is allocated or not in SDMT-1. If not we get an odd length path $(i, \nu(i), i_1, \nu(i_1), i_2, \nu(i_2))$, and so on. Finally, we will get an odd length alternating path starting from $(i, \nu(i)) = (i, a)$ w.r.t. M', which is indeed an augmenting path starting from (i, a) in SDMT-1. This concludes the proof of the induction step. $\qquad\square$

It is easy to see that both SDMT-1 and SDMT-2 reduce to SDM if all agents have strict preference over objects.

Theorem 4.4. *SDMT-2 is truthful, Pareto optimal, and terminates in $O(n_1^2\gamma + m)$ running time.*

Proof. The first two properties follow from the equivalence between SDMT-1 and SDMT-2 (Theorem 4.3) and the Pareto optimality (Corollary 3.3) and truthfulness (Theorem 3.4) of SDMT-1. It remains to establish the running time of SDMT-2.

By the previous analysis given in the proof of Theorem 3.5, in each loop iteration i, the running time is $O(|L(i)| + (i-1)\gamma)$. Summing i over $[n_1]$, we obtain that the running time of SDMT-2 is $O(m + n_1^2\gamma)$. $\qquad\square$

4.2 Randomised mechanism

We now present a universally truthful and Pareto optimal mechanism with approximation ratio of $\frac{e}{e-1}$, where agents may have weights and their preferences may involve ties (see Algorithm 3, where $e^{Y_i-1} = g(Y_i)$). Note that in the absence of agents' weights, sorting agents in the decreasing order of $w_i(1 - g(Y_i))$ simply means to sort them in the increasing order of the Y_i values, so the exponentiation is only used for the correct handling of the weights.

When preferences are strict, Algorithm 3 reduces to a variant of RSDM that has been used in weighted online bipartite matching with approximation ratio $\frac{e}{e-1}$ (see [3] and [16]). Our analysis of Algorithm 3 is a non-trivial extension of the primal-dual analysis from [16] to the case where agents' preferences may involve ties. Before analysing the approximation ratio, we will argue about the universal truthfulness of Algorithm 3 when agents' preferences are private and they in addition have weights.

If the weights are public, Algorithm 3 is universally truthful and Pareto optimal. This is because it chooses a random order of the agents, given the weights, and then runs

SDMT-2 according to this order. It follows by inspection of SDMT-2 that, if the order of the other agents is given, an agent can get a better object if he appears earlier in this order. Then it is not difficult to see that if the weights are private, and under the assumption that no agent is allowed to bid over his private weight (the so-called no-overbidding assumption – see the beginning of Section 6), Algorithm 3 is still universally truthful in the sense that no agent will lie about his preferences or weight.

Theorem 4.5. *Algorithm 3 is universally truthful, even if the weights and preference lists of the agents are their private knowledge, assuming that no agent can over-bid his weight.*

Proof. Algorithm 3 is a distribution over deterministic mechanisms due to the selection of random variables Y_i. For each deterministic mechanism (i.e., SDMT-2 when Y_i, $i \in N$ is fixed), we prove that it is truthful with respect to weights and preference lists. Let us denote by ϕ the mechanism of SDMT-2 when Y_i, $i \in N$ is fixed. If is not difficult to see that for any (W, L), $w'_i \leq w_i$ and $L'(-i)$, $i \in N$, we have $\phi_i(W, L) \succeq_i \phi_i((w'_i, w_{-i}), L) \succeq_i \phi_i((w'_i, w_{-i}), (L'(i), L(-i)))$. The first preferred order in this chain follows from the fact that the order of i when i bids w_i is better than or equal to his order when he bids w'_i. The second preferred order in this chain follows by the truthfulness of SDMT-2 when weights are public. $\square$

5 Analysis of the approximation ratio

To gain some high-level intuition behind our extension from strict preferences to preferences with ties, we highlight here the similarities and differences between our problem and that of online bipartite matching. Our problem with strict preferences and without weights is closely related to online bipartite matching.[6] If each agent in our problem ranks his desired objects in the order that precisely follows the arrival order of objects in the online bipartite matching, the two problems are equivalent. Therefore, we extend the analysis of this particular setting, where each agent's preference list is a sublist of a global preference list, to the general case where agents preferences are not constrained and may involve ties, and furthermore agents may have weights.

To analyze the approximation ratio of Algorithm 3, we first write the LP formulation of the (relaxed) problem and its dual LP formulation. Given random variables Y_i, we will define a primal solution and a dual solution obtained by Algorithm 3, which are both random variables, such that the objective value of the primal solution is always at least a fraction F of the objective value of the dual solution, and that the expectation of duals is feasible. Hence, the expectation of the primal solution is at least F times the expectation of duals, which by weak LP duality, is at least the optimal value of the primal LP. We now give the standard LP and its dual of our problem. In what follows, $G = (V, E)$, where $V = N \cup O$ and $E = \{(i, a), i \in N, a \in O\}$.

$$\max \sum_{(i,a)\in E} w_i x_{ia} \text{ such that} \qquad \min \sum_{i\in N} \alpha_i + \sum_{a\in O} \beta_a \text{ such that}$$
$$\forall i \in N : \sum_{a:(i,a)\in E} x_{ia} \leq 1 \qquad \forall (i,a) \in E : \alpha_i + \beta_a \geq w_i$$
$$\forall a \in O : \sum_{i:(i,a)\in E} x_{ia} \leq 1 \qquad \forall i \in N : \alpha_i \geq 0$$
$$\forall (i,a) \in E : x_{ia} \geq 0 \qquad \forall a \in O : \beta_a \geq 0$$

---

[6]In the online bipartite matching problem [8], vertices of one partition (think of them as agents) are given and fixed, while vertices of the other partition (think of them as objects) arrive in an adversarial order. When an item arrives, we get to see the incident edges on agents. These edges indicate the set of agents that desire this object. The algorithm must immediately match this object to one of the unmatched agents desiring it (or choose to throw it away). In the end, the size of the obtained matching is compared with the optimum matching in the realised graph.

By the next result, proved in [16], the inverse of approximation ratio is $F \in [0, 1]$.

Lemma 5.1 ([16]). *Suppose that a randomised primal-dual algorithm has a primal feasible solution with value P (which is a random variable) and a dual solution which is not necessarily feasible, with value D (which is also a random variable) such that*

1. *for some universal constant F, $P \geq F \cdot D$, always, and*

2. *the expectation of the randomised dual variables forms a feasible dual solution, that is, $\mathbb{E}(\alpha_i)$ and $\mathbb{E}(\beta_a)$ are dual feasible.*

The expectation of P is then at least $F \cdot OPT$ where OPT is the value of the optimum solution.

Proof. Since $P \geq F \cdot D$, taking expectations, $\mathbb{E}(P) \geq F \cdot \mathbb{E}(D)$. The cost of the dual solution obtained by taking expectations of the dual random variables is $\mathbb{E}(D)$ and they form a feasible dual solution, therefore $\mathbb{E}(D) \geq OPT$. Hence, $\mathbb{E}(P) \geq F \cdot OPT$. $\square$

Note that in Lemma 5.1, OPT is the weight of maximum weight matching, which is equal to the weight of a maximum weight Pareto optimal matching by Proposition 2.2. Hence, if the condition of Lemma 5.1 holds, the approximation ratio of the mechanism is $\frac{1}{F}$. The construction of the duals depends on function g. Let $F = (1 - \frac{1}{e})$. For any random selection of Y_i, $i \in N$, let $\vec{Y} = (Y_1, Y_2, \cdots, Y_{n_1}) = (Y_i, Y_{-i})$. Following the procedure of Algorithm 3, whenever agent i is matched to object a, let

$$x_{ia}(\vec{Y}) = 1, \quad \alpha_i(\vec{Y}) = w_i g(Y_i)/F, \quad \beta_a(\vec{Y}) = w_i(1 - g(Y_i))/F.$$

For all unmatched i and a, set $x_{ia}(\vec{Y}) = \alpha_i(\vec{Y}) = \beta_a(\vec{Y}) = 0$. By this definition, it follows that for any Y_i, $i \in N$, the random value P of the primal solution $\{x_{ia}(\vec{Y}), i \in N, a \in A\}$ is always identical to $F \cdot D$, where D is the random value of the dual solution $\{\alpha_i(\vec{Y}), i \in N, \beta_a(\vec{Y}), a \in O\}$.

Hence, to satisfy the conditions of Lemma 5.1, we need to show that the expectation of the dual solution $\{\alpha_i(\vec{Y}), i \in N, \beta_a(\vec{Y}), a \in O\}$ is feasible for the dual LP, implying that the approximation ratio of Algorithm 3 is at most $\frac{1}{F} = \frac{e}{e-1}$. The main technical difficulty lies in proving the dominance lemma and the monotonicity lemma (see Lemma 5.4 and 5.6). To prove these two lemmas, for any fixed agent i, and any fixed object $a \in O$, we define a threshold, denoted by $\theta = \theta_a^i$, of the random variable for Y_i, which specifies whether agent i will get matched – see Lemma 5.4. This threshold will depend on the other agents Y_{i-}. For an agent with strict preferences, such a threshold is the same as that defined in the online bipartite matching problem. However, in the presence of ties, the same defined threshold does not work. We show how to define such a threshold for our algorithm.

Let us fix an agent $i \in [n_1]$ and object $a \in O$, such that $(i, a) \in E$. Also, we fix Y_{-i}, that is, the random variables $Y_{i'}$ for all other agents $i' \neq i$. We use σ to denote the order of agents under Y_{-i}, i.e., $\sigma(1)$ is the first agent, and so on, and $\sigma([i]) = \{\sigma(1), \sigma(2), \cdots, \sigma(i)\}$. Consider Algorithm 3 running on the instance without agent i and let us denote this procedure by ALG_{-i}, where σ is the order of agents under ALG_{-i}. Given agent i and object a, the threshold $\theta = \theta_a^i$ is then defined as follows:

1. If a is unmatched in ALG_{-i}, let $\theta = 1$.

2. Otherwise, suppose that a is matched in ALG_{-i} to some agent i'. Then consider the allocations just after the "for loop" in SDMT-2 within ALG_{-i} terminated.
   If i' is labelled, set $\theta = 1$.

3. Otherwise, suppose $a \in C_j^{i'}$ and construct the trading graph $TG(i', C_j^{i'} \backslash \{a\}, [n_1] \backslash \{i\})$ from all the objects in $C_j^{i'}$ other than a (note that $\sigma([n_1 - 1]) = [n_1] \backslash \{i\}$). Recall that graph $TG(i', C_j^{i'} \backslash \{a\}, [n_1] \backslash \{i\})$ contains directed paths to all agents who can potentially provide an object for i' to trade without affecting any other agent.
   If there is a path in $TG(i', C_j^{i'} \backslash \{a\}, [n_1] \backslash \{i\})$ from i' to a labelled agent, set $\theta = 1$.

4. Otherwise, define
   $i'' = arg \min_\ell \{w_\ell(1 - g(Y_\ell))|$ there is a path from i' to ℓ in $TG(i', C_j^{i'} \backslash \{a\}, [n_1] \backslash \{i\})\}$
   Note: If index ℓ with minimum value of $w_\ell(1 - g(Y_\ell))$ is not unique, we take for i'' the largest such index. Also, observe that either $i' = i''$ or agent i' is before i'' with respect to order σ.
   If $w_i(1 - g(y)) = w_{i''}(1 - g(Y_{i''}))$ has a solution $y \in [0, 1]$ define θ to be this solution. ($g(y)$ is strictly increasing so if there is a solution, it is unique)

5. Otherwise define θ to be 0.

Now consider Algorithm 3 running on the original instance (denote such procedure as ALG), with (Y_i, Y_{-i}) fixed. Suppose that τ is the order of agents under this execution of ALG. The intuition behind the definition of θ is the following. Having Y_{-i} fixed, we want to define θ such that if we run ALG with (Y_i, Y_{-i}) where $Y_i < \theta$, then agent i gets matched. If 1. holds, then $Y_i < 1$ and i will be matched because at least object a is his available candidate. If 2. happens, then $Y_i < 1$ and i will also be matched because object a can be re-allocated from the labelled agent i' to i. Case 3. is analogous to 2. with the only difference that we now have a trading path from i' to a labelled agent. Finally, case 4. will be discussed just after Observation 5.3.

In our further analysis, we will need the following notion of a *frozen* agent or object.

Definition 5.2. *We say an agent (respectively, an object) is* frozen *if the allocation of this agent (respectively, object) remains the same until the termination of SDMT-2. We also say a trading graph is frozen if all of its agents are frozen.*

Observation 5.3. *Assume that agent i is unmatched in his turn in the "for loop" of ALG. Suppose $\tau(u) = i$, which means i selects his object in u-th iteration of the "for loop". Then at the end of the k-th iteration of the "for loop", for every $k \geq u$, there is no path from i to a labelled agent in $TG(i, L(i), \tau([k]))$, meaning this graph is frozen.*

Proof. By SDMT-2, we know that $O_u \cap L(\tau(u)) = O_u \cap L(i) = \emptyset$, which means that all the objects in $L(i)$ have been allocated to agents $\tau([u-1])$. Since $\tau(u)$ is unmatched, there is no path from $\tau(u)$ to any labelled agent in $TG(\tau(u), L(\tau(u)), \tau([u-1]))$. Let $S \subseteq \tau([u-1])$ be the set that is reachable from $\tau(u)$ in $TG(\tau(u), L(\tau(u)), \tau([u-1]))$. Clearly, each agent in S is unlabelled. Actually, notice that any agent in S is frozen. Therefore, any path through i after u-th iteration will reach an unlabelled agent. $\square$

The following two properties (dominance and monotonicity) are well known for agents with strict preference orderings. We generalise them to agents with indifferences. The difficulty of proving both dominance and monotonicity lemmas (Lemma 5.4 and 5.6) lies in case 4. (in the definition of threshold θ). This is our main technical contribution as compared to the analysis in [16].

Recall that τ (σ, resp.) is the order of agents under the execution of ALG (ALG_{-i}, resp.). We first discuss intuitions behind case 4. in the context of the Dominance Lemma (Lemma 5.4). Note that in this case there is a path from i' to i'' in $TG(i', C_j^{i'} \backslash \{a\}, [n_1] \backslash \{i\})$ and agent i'' is unlabelled. We will prove the Dominance Lemma by contradiction, using

the following two main steps. Indeed, let us assume towards a contradiction, see the text of Lemma 5.4, that $Y_i < \theta$ and i is not matched in ALG. Then the outcome of ALG is the same as that of ALG_{-i} for all the other agents (except agent i). Suppose $\sigma(u) = i''$ in ALG_{-i}, then $\tau(u+1) = i''$ in ALG under case 4. Based on the fact that outcomes of ALG and ALG_{-i} are the same (for all agents except agent i), first, we prove that either i' is labelled or there is a path, let us call it P_1, from i' to a labelled agent in $TG(i', C_j^{i'}, \tau([u]))$ at the end of the u-th iteration of the "for loop" in ALG. Secondly, due to the above property, we argue that there is a path, let us call it P_2, from i to a labelled agent in $TG(i, a, \tau([u]))$ at the end of the u-th iteration of the "for loop" in ALG, contradicting Observation 5.3; thus i will be matched. Path P_2 is constructed by the concatenation of arc (i, i') and path P_1, or the concatenation of arc (i, i'''), for some i''' on path P_1, and the rest of path P_1. The existence of P_1 is proved by a careful analysis of the structure of frozen subgraphs of the trading graph as the algorithm proceeds; the details can be found in the proof of Lemma 5.4.

Lemma 5.4 (Dominance Lemma). *Given Y_{-i}, i gets matched (to some object) if $Y_i < \theta$.*

Proof. Let us assume towards a contradiction that $Y_i < \theta$ and i is not matched in ALG. We will consider the following cases below.

Case 1. (Corresponding to case 1 in the definition of threshold θ.) If a is unmatched in ALG_{-i}, then $\theta = 1$. Suppose agent i is unmatched in ALG, then procedure ALG is the same as ALG_{-i} for all the other agents except i. But then a is always available to agent i, meaning a will be matched to agent i by process of SDMT-2, contradiction.

Case 2. If a is matched to i' in ALG_{-i}:

Case 2-(i). (Corresponding to cases 2 and 3 in the definition of threshold θ.) If i' is labelled or if there is a path from i' to a labelled agent in $TG(i', C_j^{i'} \backslash \{a\}, [n_1] \backslash \{i\})$, and i is unmatched, then there is a path from i to a labelled agent in $TG(i, a, [n_1])$. In this case, by $Trading(i, a, [n_1])$, we obtain a Pareto improvement, contradicting that SDMT-2 is Pareto optimal.

Case 2-(ii). (Corresponding to cases 5 and 4 in the definition of threshold θ.) The case $\theta = 0$ is trivial, so we consider that $w_i(1 - g(y)) = w_{i''}(1 - g(Y_{i''}))$ has a solution. Suppose that $\sigma(u) = i''$ in ALG_{-i}, then if $Y_i < \theta$, we know that $w_i(1 - g(Y_i)) > w_{i''}(1 - g(Y_{i''}))$, meaning the agent i is prior to agent i'' in ALG. Then $\tau(u + 1) = i''$ in ALG. If i is unmatched in ALG, then procedure ALG is the same as ALG_{-i} for all the other agents except i.

Suppose i'' is allocated an object b in ALG. If $i'' = i'$, then $b = a$, and if in addition $a \in O_{u+1}$, this means a is always available to all the agents prior to $\tau(u + 1) = i'' = i'$. Therefore, a will be available to i when i initially selects objects, implying that i must be allocated to some object in his turn, leading to a contradiction.

The case $a \notin O_{u+1}$ is analyzed similarly to the case $i'' \neq i'$, so we consider that $i'' \neq i'$. Since there is a path from i' to i'' after the "for loop" in ALG terminates, i is still unmatched because of our assumption towards a contradiction. Suppose that in this path $\tau(k)$ points to $\tau(u + 1) = i''$, for some $k \leq u$, then b is available to $\tau(k)$ or b has been allocated before the k-th iteration of the "for loop" in ALG. Since finally $\tau(k)$ gets an object in the same indifference class as b by Observation 4.1, before the $(u+1)$-st iteration of the "for loop" in ALG, b has been allocated by Observation 4.2. Hence, in the $(u+1)$-st iteration of the "for loop" in ALG, $\tau(u + 1)$ gets object b through the trading graph.

Observation 5.5. *The trading graph $TG(i', C_j^{i'} \backslash \{a\}, [n_1])$ after the "for loop" in ALG terminates, is exactly the same as $TG(i', C_j^{i'} \backslash \{a\}, \tau([u + 1]))$ at the end of the $(u+1)$-st iteration.*

This observation follows from the fact that otherwise, some agent $\tau(\ell)$ may be reachable from i', where $\ell > u + 1$, by process of SDMT-2, contradicting the definition of i''; note that we used here the largest index tie breaking rule.

Therefore, at the end of the u-th iteration of the "for loop" of ALG, suppose that B is the set of objects allocated to i'. Then we have the following three cases (note that ALG is the same as ALG_{-i} for all the other agents except i):

Case 2-(ii)-1. i' is labelled, then $a \in B$. Otherwise, if $a \notin B$, then in the $(u+1)$-st "for loop" iteration of ALG, a will not be allocated to i' at the end of this $(u + 1)$-st iteration by the process of SDMT-2. Thus $a \in B$, and since the trading graph $TG(i', C_j^{i'} \backslash \{a\}, [n_1])$ after the "for loop" in ALG terminates is exactly the same as $TG(i', C_j^{i'} \backslash \{a\}, \tau([u + 1]))$ at the end of the $(u+1)$-st iteration, it follows that i' will not be matched to a at the end of the "for loop" of ALG, contradiction.

Case 2-(ii)-2. i' is unlabelled and $B = \{a\}$. Then there is a path from i' to a labelled agent in $TG(i', C_j^{i'} \backslash \{a\}, \tau([u]))$. Otherwise, all the agents reachable from i' are frozen after the u-th iteration of the "for loop". This means that the allocations of those agents are fixed, because all the objects in their indifference class have been allocated by Observation 4.2. Thus, $TG(i', C_j^{i'} \backslash \{a\}, \tau([u]))$ should be the same as $TG(i', C_j^{i'} \backslash \{a\}, \tau([u + 1]))$. However, since $\tau(u + 1)$ is reachable from i' in $TG(i', C_j^{i'} \backslash \{a\}, \tau([u+1]))$, while $\tau(u + 1)$ does not appear in $TG(i', C_j^{i'} \backslash \{a\}, \tau([u]))$, we reach a contradiction.

Case 2-(ii)-3. i' is unlabelled and $B = \{c\}$, where $c \neq a$. Then there is a path from i' to a labelled agent in $TG(i', \{a\}, \tau([u]))$. Otherwise, a and the agent matched to a is frozen at the end of the u-th "for loop" iteration in ALG. This means that a will not be matched to i' at the end of the "for loop" of ALG, contradiction.

As a result, in either of the above three cases, there is a path from i to a labelled agent in $TG(i, a, \tau([u]))$ at the end of the u-th "for loop" iteration in ALG. Namely, for case **2-(ii)-1**, i points to i', which is labelled in $TG(i, a, \tau([u]))$; for case **2-(ii)-2**, i points to i' in $TG(i, a, \tau([u]))$ and there is a path from i' to a labelled agent in $TG(i', C_j^{i'} \backslash \{a\}, \tau([u])) \subseteq TG(i, a, \tau([u]))$. Finally, for case **2-(ii)-3**, suppose a is assigned to i''' at the end of the u-th "for loop" iteration, then there is a path from i''' to a labelled agent in $TG(i, a, \tau([u]))$ and i points to i''' in $TG(i, a, \tau([u]))$. This contradicts Observation 5.3. Hence, i must be matched to some object. $\square$

Let $\beta_a^s = \beta_a((s, Y_{-i}))$, when ALG denotes the execution of Algorithm 3 on the original instance and Y_{-i} is fixed and $Y_i = s$. Note that $\beta_a^\theta = w_i(1 - g(\theta))/F$. This last equality is easy to check in cases 1, 2, 3 and 5 of the definition of threshold θ. In case 4, we note that $w_i(1 - g(\theta)) = w_{i''}(1 - g(Y_{i''}))$ for some agent $i'' \neq i$. And because β_a^θ is the value of the dual variable for object a when ALG is run with $Y_i = \theta$, case 4 means that $\beta_a^\theta = w_i(1 - g(\theta))/F$, despite the fact that object a might not necessarily be assigned to agent i (however, agent i will be assigned some object).

We will now turn our attention to proving the monotonicity lemma.

Lemma 5.6 (Monotonicity Lemma). *Given Y_{-i}, for all choices of Y_i, $\beta_a^{Y_i} \geq \beta_a^\theta$.*

Before presenting the full formal proof, we will first sketch the main ideas behind the proof. The difficulty of the proof of the monotonicity lemma still lies in case 4 from the definition of threshold θ. We prove it in three steps. Recall that τ (σ, respectively) is the order of agents under the execution of ALG (ALG_{-i}, respectively). Let $\sigma(u) = i''$ in ALG_{-i}. Observe that the monotonicity lemma means that a is allocated to an agent

prior to i'' or to i''. The proof of this is easy in the case where $Y_i > \theta$. To see this, note that ALG and ALG_{-i} result in the same tentative allocation at the end of their u-th loop, since i is inserted back after i''. Hence, we only need to consider the case where $Y_i < \theta$, which implies that i is inserted back prior to i''.

- Firstly, we prove in Claim 5.7 below, that no agent, except i, is allocated a better object in ALG compared to ALG_{-i}. The argument is by contradiction: suppose there exists an agent i''' who receives a better object in ALG than in ALG_{-i}, then i must be inserted before i'''. Consequently, there exists an agent s prior to i''' who will get a worse object in ALG than in ALG_{-i}. Based on this fact, and using an alternating path argument, it is proved that there exists a path from s to i''' in s's trading graph constructed from a higher indifference class of s (than s's allocated indifference class in ALG) after i''' is allocated in ALG. This contradicts the fact that this path should not exist since the graph from that higher indifference class is frozen.

- Secondly, we prove in Claim 5.8 below, that if i' gets a worse object in ALG compared to ALG_{-i}, we prove that a must be allocated to an agent prior to i', which is in turn prior to i''. The reason is as follows: by Observation 4.2, a must be allocated and frozen before i' is considered in ALG. Then, if i' gets an object in ALG in the same indifference class as a, then we prove that there exists an agent s^* prior to i'', and suppose $\tau(u^*) = s^*$, such that there is a path from s^* to i' in $TG(s^*, C_{j^*}^{s^*}, \tau([u^*]))$ at the end of the u^*-th "for loop" iteration of ALG. Here, $C_{j^*}^{s^*}$ is the indifference class in which s^* is allocated an object in ALG_{-i}. As a consequence, by Observation 4.2, a is allocated to an agent prior to s^* and all the agents reachable from s^* in $TG(s^*, C_{j^*}^{s^*}, \tau([u^*]))$ are frozen, then a will finally be allocated to an agent prior to s^* in ALG. This means that a is allocated to an agent prior to i''.

This reasoning gives the monotonicity lemma, which together with dominance lemma is used to prove Lemma 5.9.

Proof. (full proof of the Monotonicity Lemma, Lemma 5.6)
Case 1. (Corresponding to case 1 in the definition of threshold θ.) If a is unmatched in ALG_{-i}, or if a is matched to i' in ALG_{-i} and i' is labelled, or a is matched to i' in ALG_{-i} and there is a path from i' to a labelled agent in $TG(i', C_j^{i'}\backslash\{a\}, [n_1])$, then $\theta = 1$ and $\beta_a^\theta = w_i(1 - g(\theta))/F = 0$, so $\beta_a^{Y_i} \geq \beta_a^\theta = 0$.
Case 2. (Corresponding to cases 2 and 3 in the definition of threshold θ.) If a is matched to i' in ALG_{-i}, there is no path from i' to a labelled agent in $TG(i', C_j^{i'}\backslash\{a\}, [n_1]\backslash\{i\})$. Suppose $\sigma(u) = i''$ in ALG_{-i}. Notice that $\tau([u+1]) = \sigma([u])$. Then by Observation 5.5, the trading graph $TG(i', C_j^{i'}\backslash\{a\}, \sigma([u]))$ at the end of the u-th "for loop" iteration is the same as $TG(i', C_j^{i'}\backslash\{a\}, [n_1]\backslash\{i\})$ at the termination of the "for loop" in ALG_{-i}. Otherwise, the $TG(i', C_j^{i'}\backslash\{a\}, \sigma([u]))$ is not frozen after the u-th "for loop" iteration of ALG_{-i}, meaning that there is a path from i' to a labelled agent in $TG(i', C_j^{i'}\backslash\{a\}, \sigma([u]))$. Therefore, either i' will reach an agent inferior to i'' or a labelled agent in $TG(i', C_j^{i'}\backslash\{a\}, [n_1]\backslash\{i\})$ by SDMT-2. This contradicts the definition of i''.
Case 3. (Corresponding to case 5 in the definition of threshold θ.) Suppose that equation $w_i(1 - g(y)) = w_{i''}(1 - g(Y_{i''}))$ does not have a solution, which means that $\theta = 0$ and $w_i(1 - g(Y_i))/F < w_{i''}(1 - g(Y_{i''}))/F$, for any $Y_i \in [0, 1]$. This shows that the process is the same for agents prior to agent i'' until the end of the u-th "for loop" iteration in ALG and ALG_{-i}. Since there is no path from i' to a labelled agent in $TG(i', C_j^{i'}\backslash\{a\}, \sigma([u]))$, the agents reachable from i' are frozen. Hence, a will be finally still allocated to i' in ALG, implying $\beta_a^{Y_i} = w_{i'}(1 - g(Y_{i'}))/F \geq w_{i''}(1 - g(Y_{i''}))/F > \beta_a^\theta = w_i(1 - g(0))/F$.

23

Case 4. (Corresponding to case 4 in the definition of threshold θ.) Now consider the last case that equation $w_i(1-g(y)) = w_{i''}(1-g(Y_{i''}))$ has a solution, then $\beta_a^\theta = w_i(1-g(\theta))/F = w_{i''}(1 - g(Y_{i''}))/F$. Consider the following three cases:

Case (4-i): If $Y_i > \theta$, this means $w_i(1-g(Y_i))/F < w_{i''}(1-g(Y_{i''}))/F$, and the analysis of this case is the same as above (the case that equation $w_i(1-g(y)) = w_{i''}(1-g(Y_{i''}))$ does not have a solution), since i will select objects after i''. Thus, we have $\beta_a^{Y_i} = w_{i'}(1-g(Y_{i'}))/F \geq \beta_a^\theta$.

Case (4-ii): If $Y_i < \theta$, then $w_i(1 - g(Y_i))/F > w_{i''}(1 - g(Y_{i''}))/F$, which means that i is prior to i'' in ALG. We have the following claim:

Claim 5.7. *No agent can get a better object in ALG than in ALG_{-i} after inserting i into some position from 1 to u.*

Proof. Suppose, towards a contradiction, that there exists an agent getting a better object, and let k be the smallest position where such agents are placed in ALG. Then i must be inserted in a position before k (otherwise, the process is the same for the first k agents in ALG_{-i} and ALG, so agent $\tau(k)$ can not get a better object). Let $i''' = \tau(k)$ and suppose that i''' gets an object b in ALG and object c in ALG_{-i}, where $b \succ_{i'''} c$. Observe that $\sigma(k-1) = i'''$ in ALG_{-i}. Suppose that $b \in C_j^{i'''}$ and consider the trading graph $TG(i''', C_j^{i'''}, \sigma([k-1]))$ at the end of the $(k-1)$-st "for loop" iteration of ALG_{-i}.

Let S be the set of agents reachable from i''' in $TG(i''', C_j^{i'''}, \sigma([k-1]))$ at the end of the $(k-1)$-st "for loop" iteration of ALG_{-i}. Note that any agent in S is prior to i'''. Any agent in S is allocated only one object and frozen in ALG_{-i}. Since in ALG, b is allocated to i''', then in the k-th "for loop" iteration of ALG, i''' will be allocated some objects in $C_j^{i'''}$. This means that some agent in S will get worse object compared to the allocation in ALG_{-i}.

The reason is as follows: no agent can get a better object by the definition of k. If all the agents in S can remain the same in ALG compared with ALG_{-i} (i.e., get the objects in the same indifference class in ALG and in ALG_{-i}), then the only possible allocation of S in ALG is reallocating all the objects matched to S in ALG_{-i} to S again such that each agent gets exactly one object. If there is some extra object e in ALG allocated to an agent from S in ALG, then e must be allocated to some agent j in ALG_{-i}. Since e in ALG is allocated to some agent in S, thus, j can be reached by some agent in S in ALG_{-i}. Thus, $j \in S$, which leads to a contradiction. All the objects in $C_j^{i'''}$ have been allocated to some agents in S. In ALG, we will need to allocate $|S|$ objects to $S \cup \{i'''\}$ agents because some objects owned by S in ALG_{-i} will be allocated to agent i'''. This is not possible, which gives a contradiction.

Let s be an agent in S who gets a worse object and there is a path from s to i''' in the trading graph $TG(s, d, \tau[k])$ at the end of the k-th "for loop" iteration in ALG, where d is the allocated object of s in ALG_{-i}. (Such an agent must exist: it can be found by the following procedure. Suppose $d_1 \simeq_{s_1} b$ owned by s_1 in ALG_{-i} is allocated to i''' in ALG at the end of the k-th "for loop" iteration in ALG. If s_1 gets worse in ALG compared to ALG_{-i}, then s_1 is the agent we are looking for. Otherwise, s_1 will be allocated object d_2 owned by $s_2 \in S$ in ALG_{-i} at the end of the k-th "for loop" iteration of ALG. If s_2 gets a worse object, then s_2 is the agent we are looking for. Otherwise, we continue with this procedure. By finiteness of the set S and by the fact that the agents in S own $|S|$ objects in ALG_{-i}, these objects will be allocated to agents in $S \cup \{i'''\}$ in ALG, and one of these objects will be allocated to i'''. Thus, we can find such an agent. The path from s to i''' in the trading graph $TG(s, d, \tau[k])$ at the end of the k-th "for loop" iteration in ALG is just the reverse path by the above procedure). Suppose $\tau(\ell) = s$ in ALG and $d \in C_h^s$. Consider the ℓ-th "for loop" iteration in ALG: all the agents reachable from s in

$TG(s, C_h^s, \sigma([\ell]))$ are frozen and prior to agent s since s does not obtain any object in the indifference class C_h^s.

This contradicts the fact that there is a path from s to i''' (which is inferior to s) in the trading graph $TG(s, d, \tau[k])$ at the end of the k-th "for loop" iteration in ALG. $\square$

Claim 5.8. *Object a must be allocated to an agent prior to i'' or to i'', that is, we must have $\beta_a^{Y_i} \geq w_{i''}(1 - g(Y_{i''})) = \beta_a^\theta$.*

Proof. Suppose $\sigma(u_1) = i'$ and $\sigma(u) = i''$ in ALG_{-i}. The following cases are considered:

Case (1). If i' gets worse, meaning he gets a worse object in ALG than a in ALG_{-i}, then $\tau(u_1 + 1) = i'$ in ALG (i is inserted back prior to i'). Thus, all the agents reachable from i' in $TG(i', a, \tau([u_1 + 1]))$ are frozen and the agent who owns a will finally get a. This agent is prior to i', giving that $\beta_a^{Y_i} \geq w_{i'}(1 - g(Y_{i'})) \geq w_{i''}(1 - g(Y_{i''})) = \beta_a^\theta$.

Case (2). If i' gets a in ALG, then we are done. Otherwise, suppose i' gets an object $a' \simeq_{i'} a$, $a', a \in C_j^{i'}$ in ALG. Denote by S^* the set of agents reachable from i' in $TG(i', C_j^{i'} \backslash \{a\}, \sigma([n_1 - 1]))$ at the end of the "for loop" of ALG_{-i} (note that $\sigma([n_1 - 1]) = [n_1] \backslash \{i\}$). If no one in S^* gets worse in ALG than in ALG_{-i}, then a must be allocated to some agent in S^*. The reason is similar to the above argument. All agents in S^* get exactly one object. If a is not allocated in S^*, no one gets worse in S^*, and there must be an extra object b allocated to some agent j in S^*. No matter whom b is allocated to in ALG_{-i}, there is a path from j to this agent. Hence, this agent belongs to S^*, a contradiction. Note that, by the definition of i'', for any $s \in S^*$, $\sigma^{-1}(s) > \sigma^{-1}(i'')$ ($\sigma^{-1}(s)$ denotes the order of s in σ or in ALG_{-i}) implies that $w_s(1 - g(Y_s)) \geq w_{i''}(1 - g(Y_{i''}))$. Therefore $\beta_a^{Y_i} \geq w_{i''}(1 - g(Y_{i''})) = \beta_a^\theta$.

Otherwise, by the previous argument, there exists $s^* \in S^*$ who gets worse in ALG compared to ALG_{-i}, and there is a path from $s^* \in S^*$ to i' in $TG(s^*, d^*, [n_1])$ at the end of ALG, where d^* is the allocation of s^* in ALG_{-i}. If s^* is prior to i', then by similar argument as above, there should be no path from s^* to an agent inferior to s^* (constructed from the objects in $L(s^*)$ no worse than d^*) at the end of ALG, a contradiction. Hence, s^* can only be inferior to i'. Suppose $\tau(u^*) = s^*$ and $d^* \in C_{j^*}^{s^*}$, then we know that there is a path from s^* to i' in $TG(s^*, C_{j^*}^{s^*}, [n_1])$ at the end of the "for loop" of ALG. Next we will prove the following statement (which we denote as $(*)$):

There is a path from s^* to i' in $TG(s^*, C_{j^*}^{s^*}, \tau([u^*]))$ at the end of u^*-th "for loop" of ALG.

If $(*)$ is true, then all the agents reachable from s^* in $TG(s^*, C_{j^*}^{s^*}, \tau([u^*]))$ are frozen, and a is allocated to an agent prior to s^* due to Observation 4.2. Then, we have that $\beta_a^{Y_i} \geq w_{s^*}(1 - g(Y_{s^*})) \geq w_{i''}(1 - g(Y_{i''})) = \beta_a^\theta$ since $s^* \in S^*$.

Suppose finally that $(*)$ is not true, then all the agents U^* that are reachable from s^* in $TG(s^*, d^*, \tau([u^*]))$ have been frozen. U^* will remain the same until the end of ALG and $i' \notin U^*$. However, by the definition of S^* and by $s^* \in S^*$, there is a path from s^* to i' in $TG(s^*, d^*, [n_1])$ at the end of ALG, meaning $i' \in U^*$, a contradiction. $\square$

By Claim 5.8, we know that if $Y_i < \theta$ then $\beta_a^{Y_i} \geq w_{i''}(1 - g(Y_{i''}))/F = \beta_a^\theta$.

Case (4-iii): If $Y_i = \theta$, this means that $w_i(1 - g(Y_i))/F = w_{i''}(1 - g(Y_{i''}))/F$. If $i'' < i$, the case is same as if $Y_i > \theta$. Otherwise, it falls into the case $Y_i < \theta$.

To summarise, for all choices of Y_i, $\beta_a^{Y_i} \geq \beta_a^\theta$. $\square$

Let us recall that $F = (1 - \frac{1}{e})$ and $g(y) = e^{y-1}$. Observe that $\int g(y)dy = g(y) + C$, where C is any fixed constant. Then it is not difficult to see that

$$\text{for each } t \in [0, 1] : \int_0^t g(y)dy + 1 - g(t) = F \tag{1}$$

25

Lemma 5.9 ([16]). $\forall (i, a) \in E$, $\mathbb{E}_{\vec{Y}}(\alpha_i(\vec{Y}) + \beta_a(\vec{Y})) \geq w_i$.

Proof. For fixed choices of Y_{-i}, by the Dominance Lemma (Lemma 5.4), i is matched whenever $Y_i < \theta$. Hence,

$$\mathbb{E}_{Y_i}(\alpha_i(\vec{Y})) \geq w_i \int_0^\theta g(y) dy / F.$$

By the Monotonicity Lemma (Lemma 5.6), $\beta_a(\vec{Y}) = \beta_a^{Y_i} \geq \beta_a^\theta = w_i(1 - g(\theta))/F$, for any $Y_i \in [0, 1]$, then

$$\mathbb{E}_{Y_i}(\beta_a(\vec{Y})) \geq w_i(1 - g(\theta))/F.$$

Therefore, note that by formula (1), we have

$$\mathbb{E}_{Y_i}(\alpha_i(\vec{Y}) + \beta_a(\vec{Y})) \geq w_i \int_0^\theta g(y) dy / F + w_i(1 - g(\theta))/F = w_i.$$

As a result, $\mathbb{E}_{\vec{Y}}(\alpha_i(\vec{Y}) + \beta_a(\vec{Y})) \geq w_i$. $\qquad\square$

From Lemma 5.1 and Lemma 5.9, we have the following theorem.

Theorem 5.10. *Algorithm 3 achieves an approximation ratio of $\frac{e}{e-1}$ for weighted agents with indifferences.*

6   Online interpretation and lower bounds

We will first provide here an "online" flavour interpretation of the weighted version of our problem. We interpret it in the following way. An administrator holds all the objects, and all agents with unknown preference lists are applicants for these objects. We assume that weights are private information of each agent, but that they cannot overstate their weights, a so-called no-overbidding assumption. Applicants are interviewed one-by-one in a random order. A decision about each particular applicant is to be made immediately after the interview. During the interview, the applicant selects his favourite object among the available remaining objects if there exists one in his preference list and must be allocated (matched to) that object because we consider only truthful mechanisms.[7] This applicant will not be interviewed again. The administrator can know the number of matched applicants interviewed so far, but is unaware whether yet unseen applicants will be matched or not. Our goal is to find the optimal strategy, that is a (random) arrival order of agents that maximises the ratio between the total weight of matched agents and the maximum weight of a matching if all the agents preference lists are known in advance.

We will now describe the required preliminaries that will be used in the remainder of this section to prove the lower bounds.

Preliminaries. We will use Yao's minmax principle, see [25, Proposition 2.5 (page 35)] and [37], to obtain a non-trivial lower bound for universally truthful and Pareto optimal

---

[7]We can extend this setting to the case where the administrator can decide whether to let the applicant select his favourite object or to reject this applicant, meaning that the applicant gets nothing. In this more general problem, it is not difficult to prove that for any fixed order of the applicants, the decision that the administrator does not reject any applicant will maximise the number of matched applicants. Therefore, this more general problem is reduced to the setting where the administrator lets each applicant select his favourite object, and hence our lower bound from Section 6 also applies to this setting.

mechanisms and another lower bound for an "online" version of our problem. We first need some preliminaries.

Let us fix the number of agents n_1 and the number of objects n_2. The number of distinct instances and the number of deterministic truthful and Pareto optimal mechanisms are finite. Denote by $\mathcal{T}$ the set of deterministic truthful and Pareto optimal mechanisms with input size n_1 and n_2, and $\mathcal{I}$ the set of instances with input size n_1 and n_2. Let P and Q denote the set of probability distributions on $\mathcal{T}$ and $\mathcal{I}$, respectively. Denote $\mathbb{E}_{p,q}(r(T_p, I_q))$ as the inverse of approximation ratio when the input I_q is sampled according to the distribution $q \in$ Q and a universally truthful and Pareto optimal mechanism T_p is sampled according to the distribution $p \in$ P. Then the minmax theorem [37] states the following:

$$\min_{q \in Q} \max_{p \in P} \mathbb{E}_{p,q}(r(T_p, I_q)) = \max_{p \in P} \min_{q \in Q} \mathbb{E}_{p,q}(r(T_p, I_q))$$

and

$$\min_{q \in Q} \max_{T \in \mathcal{T}} \mathbb{E}_q(r(T, I_q)) = \max_{p \in P} \min_{I \in \mathcal{I}} \mathbb{E}_p(r(T_p, I)).$$

As a consequence, for any $q \in$ Q and $p \in$ P, we have

$$\max_{T \in \mathcal{T}} \mathbb{E}_q(r(T, I_q)) \geq \min_{I \in \mathcal{I}} \mathbb{E}_p(r(T_p, I)).$$

This inequality states that an upper bound on the inverse of the approximation ratio of the best universally truthful and Pareto optimal mechanism T_p on the worst instance is upper bounded by the inverse of the approximation ratio of the best deterministic truthful and Pareto optimal mechanism on a randomly chosen instance. Hence, in order to bound $\min_{I \in \mathcal{I}} \mathbb{E}_p(r(T_p, I))$, we only need to construct an appropriate random instance and compute the upper bound of the best deterministic truthful and Pareto optimal mechanism on this random instance. Consider the *triangle instance* where $N = \{1, 2, \cdots, n_1\}$ and $O = \{o_1, o_2, \cdots, o_{n_1}\}$, and an agent i's preference ordering is $o_1 \succ_i o_2 \succ_i \cdots \succ_i o_i$, for any $i \in N$.

Let $\mathcal{S}$ denote the set of all the permutations of agents' preference lists of the triangle instance. Consider now a random instance S_{uni} as the uniform distribution of $\mathcal{S}$. It is obvious that the output of any serial dictatorship mechanism (which is a deterministic, truthful and Pareto optimal mechanism, defined by a specific fixed order of the agents) running on $\mathcal{S}$ is the same. Hence, for any serial dictatorship mechanism (SDM), $\mathbb{E}_{uni}(r(SDM, S_{uni}))$ is equal to the inverse of the approximation ratio of RSDM, which is just SDM with the order of agents chosen uniformly at random, when running on the triangle instance.

Online lower bound. We now apply these preliminaries to the online version of our problem. Recall that applicants in this online problem are truthful due to the truthfulness of serial dictatorship mechanism. The strategy of the administrator is a random order in which the applicants are interviewed. More precisely, let Π denote the set of all the permutations of applicants and $P(\Pi)$ be the set of probability distributions on Π. Let Π_p be a random order of applicants, where the order is selected according to the distribution $p \in P(\Pi)$ on Π, and then the strategy set of the administrator is $\{\Pi_p : p \in P(\Pi)\}$. We will show that the best strategy for the administrator is to select applicants' order uniformly at random.

Theorem 6.1. *The best strategy for the administrator in the online problem is to select the applicants' order uniformly at random. Thus, any other randomised strategy, than the one used in Algorithm 3, would lead to an approximation guarantee worse than $\frac{e}{e-1}$.*

Proof. This proof is similar to the classical proof from [22]. In particular it uses the same class of instances. Let $\mathbb{E}_{p,q}(r(\Pi_p, I_q))$ be the inverse of the approximation ratio when the random order is Π_p and the random instance is I_q, and let Π_{uni} denote the uniform order. By the approximation ratio of RSDM, for any I, $\mathbb{E}_{uni}(r(\Pi_{uni}, I)) \geq \frac{e-1}{e}$. Now for upper bound of $\mathbb{E}_{p,q}(r(\Pi_p, I_q))$, by Yao's principle [25, Proposition 2.5], $\max_{T \in \Pi} \mathbb{E}_q(r(T, I_q)) \geq \min_{I \in \mathcal{I}} \mathbb{E}_p(r(T_p, I))$. Recall that S_{uni} is the uniform distribution over $\mathcal{S}$. Then we need to upper bound $\max_{T \in \Pi} \mathbb{E}_q(r(T, S_{uni}))$, which in fact is equal to the inverse of the approximation ratio obtained by running RSDM on the triangle instance, which is $\frac{e-1}{e}$. The argument is as follows. Suppose object o_k is allocated by RSDM with probability $p_k \leq 1$ on the triangle instance. Then, because there are $n_1 - k + 1$ agents with o_k in their preference lists, each such agent obtains o_k with equal probability $\frac{p_k}{n_1-k+1}$. Therefore, agent i is allocated an object with probability $\sum_{j=1}^{i} \frac{p_j}{n_1-j+1}$, which is at most $\min\{1, \sum_{j=1}^{i} \frac{1}{n_1-j+1}\}$. Now, summing over all the agents, by a simple calculation we get that the expected cardinality of the number of allocated agents is at most $n_1(1 - \frac{1}{e})$, for large enough n_1. Hence, the approximation ratio is tight. $\square$

Lower bound for randomised mechanisms. If we can prove that the output of any deterministic truthful and Pareto optimal mechanism running on $\mathcal{S}$ is the same as that of SDM then $\max_{T \in \mathcal{T}} \mathbb{E}_q(r(T, I)) = 1 - \frac{1}{e}$. To show our lower bound it suffices to show that the sum of the sizes of all the matchings returned by any deterministic truthful and Pareto optimal mechanism executed on $\mathcal{S}$ is smaller than that of returned by any SDM executed on $\mathcal{S}$. Then $\max_{T \in \mathcal{T}} \mathbb{E}_q(r(T, I)) = 1 - \frac{1}{e}$. We use $\#^\phi(\mathcal{S})$ to denote the sum of the sizes of all the matchings returned by mechanism ϕ when executed on $\mathcal{S}$. We want to to prove that $\#^\phi(\mathcal{S}) \leq \#^{SDM}(\mathcal{S})$, for any n_1 and n_2 and for any universally truthful and Pareto optimal mechanism ϕ. We can prove this inequality assuming $n_1 = n_2 = 3$, which gives us the lower bound of $\frac{18}{13}$ for any universally truthful and Pareto optimal mechanism.

Theorem 6.2. *For any deterministic truthful and Pareto optimal mechanism ϕ, $\#^\phi(\mathcal{S}) \leq 13$, when $n_1 = 3$. Thus, any universally truthful and Pareto optimal mechanism for this problem has an approximation ratio of at least $\frac{18}{13}$.*

Proof. Suppose the agents are 1, 2, 3 and objects are a, b, c. We use the notation $\begin{pmatrix} \underline{a} & b & c \\ a & \underline{b} & \\ a & b & \underline{c} \end{pmatrix}$ to denote assignments that allocate a to agent 1, b to agent 2 and c to agent 3, where row i denotes agent i's preference list and preference ordering is the increasing order of column indices, $i = 1, 2, 3$. If there are no underlines of the objects, then this notation denotes the input of mechanism. Note that in this setting, $\mathcal{S} = \left\{ \begin{pmatrix} a & & \\ a & b & \\ a & b & c \end{pmatrix}, \right.$

$\begin{pmatrix} a & & \\ a & b & c \\ a & b & \end{pmatrix}, \begin{pmatrix} a & b & \\ a & & \\ a & b & c \end{pmatrix}, \begin{pmatrix} a & b & \\ a & b & c \\ a & & \end{pmatrix}, \begin{pmatrix} a & b & c \\ a & & \\ a & b & \end{pmatrix}, \left. \begin{pmatrix} a & b & c \\ a & b & \\ a & & \end{pmatrix} \right\}$. We would like to show that for any deterministic truthful and Pareto optimal mechanism ϕ, $\#^\phi(\mathcal{S}) \leq 13$. Without loss of generality, suppose $\begin{pmatrix} \underline{a} & b & c \\ a & \underline{b} & c \\ a & b & \underline{c} \end{pmatrix}$, and we will consider the following two cases:

Case (i): If $\begin{pmatrix} a & b & \underline{c} \\ a & \underline{b} & \\ \underline{a} & b & c \end{pmatrix}$, then we will show that $\begin{pmatrix} \underline{a} & b & \\ a & \underline{b} & c \\ a & b & \underline{c} \end{pmatrix}$. (Observe that the first agent must get a because otherwise we have contradiction with truthfulness by

$$\begin{pmatrix} \underline{a} & b & c \\ a & \underline{b} & c \\ a & b & \underline{c} \end{pmatrix} .)$$

Now, if $\begin{pmatrix} \underline{a} & b & \\ a & \underline{b} & c \\ a & b & \underline{c} \end{pmatrix}$ would not hold then $\begin{pmatrix} \underline{a} & b & \\ a & b & \underline{c} \\ a & \underline{b} & c \end{pmatrix}$. Then we obtain $\begin{pmatrix} a & b & \\ a & b & \\ \underline{a} & b & c \end{pmatrix}$.

The reason is as follows: $\begin{pmatrix} a & b & \underline{c} \\ a & \underline{b} & \\ \underline{a} & b & c \end{pmatrix}$ implies that the first agent in the input $\begin{pmatrix} a & b & \\ a & b & \\ a & b & c \end{pmatrix}$ cannot get any object by truthfulness. Similarly, from $\begin{pmatrix} \underline{a} & b & \\ a & b & \underline{c} \\ a & \underline{b} & c \end{pmatrix}$, the second agent in $\begin{pmatrix} a & b & \\ a & b & \\ a & b & c \end{pmatrix}$ cannot get any object by truthfulness. Thus we have that $\begin{pmatrix} a & b & \\ a & b & \\ \underline{a} & b & c \end{pmatrix}$, which is a contradiction to Pareto optimality.

By a similar argument, we have $\begin{pmatrix} a & b & \\ a & \underline{b} & \\ \underline{a} & b & c \end{pmatrix}$, $\begin{pmatrix} \underline{a} & & \\ a & \underline{b} & c \\ a & b & \underline{c} \end{pmatrix}$ and $\begin{pmatrix} a & & \\ a & \underline{b} & \\ \underline{a} & b & c \end{pmatrix}$. From $\begin{pmatrix} a & b & \\ a & \underline{b} & \\ \underline{a} & b & c \end{pmatrix}$, we know the size of the matching output from $\begin{pmatrix} a & b & \\ a & & \\ a & b & c \end{pmatrix}$ is 2. From $\begin{pmatrix} a & b & \underline{c} \\ a & \underline{b} & \\ \underline{a} & b & c \end{pmatrix}$, we know the size of the matching output from $\begin{pmatrix} a & b & c \\ a & & \\ a & b & \end{pmatrix}$ is 2. From $\begin{pmatrix} \underline{a} & b & \\ a & \underline{b} & c \\ a & b & \underline{c} \end{pmatrix}$, we know the size of the matching output from $\begin{pmatrix} a & b & \\ a & b & c \\ a & & \end{pmatrix}$ is 2. From $\begin{pmatrix} \underline{a} & & \\ a & \underline{b} & c \\ a & b & \underline{c} \end{pmatrix}$, we know the size of the matching output from $\begin{pmatrix} a & & \\ a & b & c \\ a & b & \end{pmatrix}$ is at most 2.

Thus, if the current mechanism is ϕ^1 then $\#^{\phi^1}(\mathcal{S}) \leq 13$.

Case (ii): If $\begin{pmatrix} \underline{a} & b & c \\ a & \underline{b} & \\ a & b & \underline{c} \end{pmatrix}$, we consider the following two cases:

Case (ii-a): If $\begin{pmatrix} \underline{a} & b & \\ a & b & \underline{c} \\ a & \underline{b} & c \end{pmatrix}$, then $\begin{pmatrix} \underline{a} & b & \\ a & b & \\ a & \underline{b} & c \end{pmatrix}$, and we conclude that $\begin{pmatrix} \underline{a} & b & c \\ a & \underline{b} & c \\ a & b & \end{pmatrix}$, otherwise suppose $\begin{pmatrix} a & \underline{b} & c \\ \underline{a} & b & c \\ a & b & \end{pmatrix}$ (since $\begin{pmatrix} \underline{a} & b & c \\ a & \underline{b} & c \\ a & b & \underline{c} \end{pmatrix}$), then b is allocated to agent 1 in $\begin{pmatrix} a & b & \\ a & b & c \\ a & b & \end{pmatrix}$. From $\begin{pmatrix} \underline{a} & b & \\ a & b & \underline{c} \\ a & \underline{b} & c \end{pmatrix}$, we know b is allocated to agent 3 in $\begin{pmatrix} a & b & \\ a & b & c \\ a & b & \end{pmatrix}$, a contradiction. Hence, $\begin{pmatrix} \underline{a} & b & c \\ a & \underline{b} & c \\ a & b & \end{pmatrix}$, then we know $\begin{pmatrix} \underline{a} & b & c \\ a & \underline{b} & \\ a & b & \end{pmatrix}$ (from $\begin{pmatrix} \underline{a} & b & c \\ a & \underline{b} & c \\ a & b & \end{pmatrix}$ and $\begin{pmatrix} \underline{a} & b & c \\ a & \underline{b} & \\ a & b & \underline{c} \end{pmatrix}$), and $\begin{pmatrix} \underline{a} & b & \\ a & b & \underline{c} \\ a & \underline{b} & \end{pmatrix}$ (From $\begin{pmatrix} \underline{a} & b & c \\ a & \underline{b} & c \\ a & b & \end{pmatrix}$ and $\begin{pmatrix} \underline{a} & b & \\ a & b & \underline{c} \\ a & \underline{b} & c \end{pmatrix}$). Now the

matching size of assignment of $\begin{pmatrix} a & b & c \\ a & & \\ a & b & \end{pmatrix}$ and $\begin{pmatrix} a & b & c \\ a & b & \\ a & & \end{pmatrix}$ is both 2 (from $\begin{pmatrix} \underline{a} & b & c \\ a & \underline{b} & \\ a & b & \end{pmatrix}$).

The matching size of assignment of $\begin{pmatrix} a & b & \\ a & b & c \\ a & & \end{pmatrix}$ is 2 since $\begin{pmatrix} \underline{a} & b & \\ a & b & \underline{c} \\ a & \underline{b} & \end{pmatrix}$. The matching

size of assignment of $\begin{pmatrix} a & b & \\ a & & \\ a & b & c \end{pmatrix}$ is 2 following from $\begin{pmatrix} \underline{a} & b & \\ a & b & \\ a & \underline{b} & c \end{pmatrix}$. Consider the assign-

ment of $\begin{pmatrix} a & & \\ a & b & c \\ a & b & c \end{pmatrix}$, no matter what the assignment is, at most one matching size of

assignment of $\begin{pmatrix} a & & \\ a & b & \\ a & b & c \end{pmatrix}$ and $\begin{pmatrix} a & & \\ a & b & c \\ a & b & \end{pmatrix}$ is 3. Denote the mechanism in this case by

ϕ_2, then $\#^{\phi_2}(\mathcal{S}) \leq 13$.

Case (ii-b): If $\begin{pmatrix} \underline{a} & b & \\ a & \underline{b} & c \\ a & b & \underline{c} \end{pmatrix}$, recall that $\begin{pmatrix} \underline{a} & b & c \\ a & \underline{b} & \\ a & b & \underline{c} \end{pmatrix}$ and $\begin{pmatrix} \underline{a} & b & c \\ a & \underline{b} & c \\ a & b & \underline{c} \end{pmatrix}$, consider the

following two cases:

Case (ii-b-1): If $\begin{pmatrix} \underline{a} & b & c \\ a & \underline{b} & c \\ a & b & \end{pmatrix}$, then $\begin{pmatrix} \underline{a} & b & c \\ a & \underline{b} & \\ a & b & \end{pmatrix}$ since $\begin{pmatrix} \underline{a} & b & c \\ a & \underline{b} & \\ a & b & \underline{c} \end{pmatrix}$. We know the

matching sizes of assignment of $\begin{pmatrix} a & b & c \\ a & b & \\ a & & \end{pmatrix}$ and $\begin{pmatrix} a & b & c \\ a & & \\ a & b & \end{pmatrix}$ are both 2. Since $\begin{pmatrix} \underline{a} & b & \\ a & \underline{b} & \\ a & b & \underline{c} \end{pmatrix}$

due to $\begin{pmatrix} \underline{a} & b & c \\ a & \underline{b} & \\ a & b & \underline{c} \end{pmatrix}$ and $\begin{pmatrix} \underline{a} & b & c \\ a & \underline{b} & c \\ a & b & \end{pmatrix}$, the matching size of assignment of $\begin{pmatrix} a & b & \\ a & & \\ a & b & c \end{pmatrix}$

is 2. Since $\begin{pmatrix} \underline{a} & b & \\ a & \underline{b} & c \\ a & b & \end{pmatrix}$ due to $\begin{pmatrix} \underline{a} & b & c \\ a & \underline{b} & c \\ a & b & \end{pmatrix}$ and $\begin{pmatrix} \underline{a} & b & \\ a & \underline{b} & c \\ a & b & \underline{c} \end{pmatrix}$, then the matching size

of assignment of $\begin{pmatrix} a & b & \\ a & b & c \\ a & & \end{pmatrix}$ is 2. Similar as the above argument, consider the assign-

ment of $\begin{pmatrix} a & & \\ a & b & c \\ a & b & c \end{pmatrix}$, no matter what the assignment is, at most one matching size of

assignment of $\begin{pmatrix} a & & \\ a & b & \\ a & b & c \end{pmatrix}$ and $\begin{pmatrix} a & & \\ a & b & c \\ a & b & \end{pmatrix}$ is 3. Denote the mechanism in this case by

ϕ_3, then $\#^{\phi_3}(\mathcal{S}) \leq 13$.

Case (ii-b-2): If $\begin{pmatrix} a & \underline{b} & c \\ \underline{a} & b & c \\ a & b & \end{pmatrix}$, recall that we have $\begin{pmatrix} \underline{a} & b & \\ a & \underline{b} & c \\ a & b & \underline{c} \end{pmatrix}$, $\begin{pmatrix} \underline{a} & b & c \\ a & \underline{b} & \\ a & b & \underline{c} \end{pmatrix}$ and

$\begin{pmatrix} \underline{a} & b & c \\ a & \underline{b} & c \\ a & b & \underline{c} \end{pmatrix}$. From $\begin{pmatrix} a & \underline{b} & c \\ \underline{a} & b & c \\ a & b & \end{pmatrix}$ and $\begin{pmatrix} \underline{a} & b & \\ a & \underline{b} & c \\ a & b & \underline{c} \end{pmatrix}$, we get $\begin{pmatrix} a & \underline{b} & \\ \underline{a} & b & c \\ a & b & \end{pmatrix}$, then the

matching sizes of assignment of $\begin{pmatrix} a & & \\ a & b & c \\ a & b & \end{pmatrix}$ and $\begin{pmatrix} a & b & \\ a & b & c \\ a & & \end{pmatrix}$ are both 2. From $\begin{pmatrix} \underline{a} & b & c \\ a & \underline{b} & \\ a & b & \underline{c} \end{pmatrix}$

and $\begin{pmatrix} \underline{a} & b & \\ a & \underline{b} & c \\ a & b & \underline{c} \end{pmatrix}$, we get $\begin{pmatrix} \underline{a} & b & \\ a & \underline{b} & \\ a & b & \underline{c} \end{pmatrix}$, then the matching size of assignment of $\begin{pmatrix} a & b & \\ a & & \\ a & b & c \end{pmatrix}$

is 2. From $\begin{pmatrix} \underline{a} & b & c \\ a & \underline{b} & \\ a & b & \underline{c} \end{pmatrix}$ and $\begin{pmatrix} a & \underline{b} & c \\ \underline{a} & b & c \\ a & b & \end{pmatrix}$, we get $\begin{pmatrix} a & \underline{b} & c \\ \underline{a} & b & \\ a & b & \end{pmatrix}$, then the matching size

of assignment of $\begin{pmatrix} a & b & c \\ a & b & \\ a & & \end{pmatrix}$ is 2. From $\begin{pmatrix} a & \underline{b} & c \\ \underline{a} & b & \\ a & b & \end{pmatrix}$ and $\begin{pmatrix} \underline{a} & b & \\ a & \underline{b} & \\ a & b & \underline{c} \end{pmatrix}$, it follows that

$\begin{pmatrix} a & \underline{b} & \\ \underline{a} & b & \\ a & b & \end{pmatrix}$, we conclude $\begin{pmatrix} \underline{a} & & \\ a & \underline{b} & \\ a & b & \underline{c} \end{pmatrix}$ is not true. Otherwise from $\begin{pmatrix} \underline{a} & & \\ a & \underline{b} & \\ a & b & \underline{c} \end{pmatrix}$ and

$\begin{pmatrix} a & \underline{b} & \\ \underline{a} & b & \\ a & b & \end{pmatrix}$, it follows that $\begin{pmatrix} a & & \\ \underline{a} & b & \\ a & b & \end{pmatrix}$, which contradicts to the Pareto optimality of

the mechanism. Hence, the matching size of assignment of $\begin{pmatrix} a & & \\ a & b & \\ a & b & c \end{pmatrix}$ is 2. It is obvi-

ous to see that the matching size of assignment of $\begin{pmatrix} a & b & c \\ a & & \\ a & b & \end{pmatrix}$ is at most 3. Denote the

current mechanism as ϕ^4, we know that $\#^{\phi^4}(\mathcal{S}) \leq 13$. $\qquad\square$

Note that Theorem 6.2 shows that $\min_{I\in\mathcal{I}} \mathbb{E}_p(r(T_p, I)) \leq \max_{T\in\mathcal{T}} \mathbb{E}_q(r(T, S_{uni})) \leq \frac{13}{18}$, for any $p \in \mathrm{P}$, and when S_{uni} is the uniform distribution over $\mathcal{S}$. Hence, the approximation ratio is at least $\frac{18}{13}$.

Lower bound for non-bossy mechanisms. In this subsection we only consider the unweighted HA problem and with strict preferences. Thus an instance of HA is just $I = (N, O, L)$, where $L = (L(1), \dots, L(n_1))$ is the joint list of (strict) preferences of the agents.

We first define the concept of non-bossiness for a deterministic mechanism (see, e.g., [26]). A deterministic mechanism ϕ is non-bossy if for any agent $i \in N$, any joint preference list profile L, and any preference list $L'(i)$ of agent i, if $\phi_i(L(i), L(-i)) = \phi_i(L'(i), L(-i))$ then $\phi(L(i), L(-i)) = \phi(L'(i), L(-i))$. Roughly speaking, non-bossiness ensures that no agent can change the allocation of other agents, by reporting a different preference list, without changing his own allocation.

Bade [7] showed that (Theorem 1 in [7]) any mechanism that is truthful, Pareto optimal and non-bossy is *s-equivalent* to SDM, in the sense that if the order of agents is generated uniformly at random, the matching returned by SDM is the same as the one returned by any truthful, Pareto optimal and non-bossy mechanism.

We will now briefly introduce the required notions from [7] to be able to formally use the result of Bade [7]. Let $\phi : \mathcal{I} \to \mathcal{M}$ be any deterministic mechanism for HA and σ be any order (permutation) of the agents. We define a *permuted mechanism* $\sigma \odot \phi : \mathcal{I} \to \mathcal{M}$ via $(\sigma \odot \phi)_i(L) = \phi_{\sigma^{-1}(i)}(L(\sigma(1)), \dots, L(\sigma(n_1)))$ for any agent $i \in N$. Intuitively, permutation σ assigns each agent in N to a role in the mechanism, such that the agent $\sigma(i)$ under $\sigma \odot \phi$ assumes the role that agent i plays under ϕ.

The *symmetrization* of a deterministic mechanism $\phi : \mathcal{I} \to \mathcal{M}$ is a random mechanism $Rand(\phi) : \mathcal{I} \to Rand(\mathcal{M})$ that calculates the probability of matching μ at the joint preferences list L as the probability of a permutation σ with $\mu = (\sigma \odot \phi)(L)$ under the uniform distribution on $\Pi(N)$, i.e., where $\sigma \in \Pi(N)$ is a uniform random permutation of the agents, and $\Pi(N)$ is the set of all permutations of the agents. So we have:

$$\mathbf{Pr}\left[Rand(\phi)(L) = \mu\right] = \frac{|\{\sigma : (\sigma \odot \phi)(L) = \mu\}|}{n_1!}.$$

For instance a symmetrization of SDM is simply RSDM.

We say that two deterministic mechanisms ϕ and ϕ' are *s-equivalent* if $Rand(\phi) = Rand(\phi')$. The main result of Bade [7] can now be stated as.

Theorem 6.3 (Theorem 1 in [7])**.** *Any (deterministic) truthful, Pareto optimal and non-bossy mechanism for HA is s-equivalent to serial dictatorship mechanism (SDM).*

Using this theorem we can now prove the following tight lower bound.

Theorem 6.4. *No random mechanism which is a symmetrization of any truthful, non-bossy and Pareto optimal mechanism can achieve the approximation ratio better than $\frac{e}{e-1}$.*

Proof. Let ϕ be any deterministic truthful, non-bossy and Pareto optimal mechanism for the HA problem with strict preferences. By Theorem 6.3 the random mechanism $Rand(\phi)$ is equivalent to RSDM. In the proof of Theorem 6.1, we have shown that the expected aproximation ratio of RSDM cannot be better than $\frac{e}{e-1}$ on the triangle instances of HA for large enough n_1. This concludes the argument. $\square$

Note that our mechanism with strict preference lists and weights is a symmetrization of a truthful, non-bossy and Pareto optimal mechanism SDMT-2.

7 Conclusion

Whilst this paper has focused on Pareto optimality in the HA context, stronger forms of optimality are possible. For example, *minimum cost* (or *maximum utility*), *rank-maximal* and *popular* matchings can also be studied in the HA context, and a matching of each of these types is Pareto optimal (see, e.g., [24, Sec. 1.5] for definitions). As Pareto optimality is a unifying feature of all of these other forms of optimality, we chose to concentrate on this concept in our search for randomised truthful mechanisms that can provide good approximations to maximum matchings with desirable properties. Note that the lower bound on the performance of deterministic truthful mechanisms that produce Pareto optimal matchigns extends to those producing matchings that satisfy these stronger optimality criteria. It will thus be the focus of future work to consider the performance of randomised truthful mechanisms for these problems.

As far as lower bounds for randomised Pareto optimal mechanisms are concerned, we proved a lower bound of $\frac{18}{13}$ for any universally truthful Pareto optimal mechanism. Moreover, we obtained a tight lower bound for the class of symmetrizations of truthful, Pareto optimal and non-bossy mechanisms using a characterization due to Bade [7]. We believe that the existence of a lower bound of $\frac{e}{e-1}$ for any universally truthful Pareto optimal mechanism is an interesting open question, and our lower bound of $\frac{18}{13}$ is a useful step towards resolving this.

Acknowledgements

References

[1] A. Abdulkadiroğlu and T. Sönmez. Random serial dictatorship and the core from random endowments in house allocation problems. *Econometrica*, 66(3):689–701, 1998.

[2] D.J. Abraham, K. Cechlárová, D.F. Manlove, and K. Mehlhorn. Pareto optimality in house allocation problems. In *Proceedings of ISAAC '04: the 15th Annual International Symposium on Algorithms and Computation*, volume 3341 of *Lecture Notes in Computer Science*, pages 3–15. Springer, 2004.

[3] G. Aggarwal, G. Goel, C. Karande, and A. Mehta. Online vertex-weighted bipartite matching and single-bid budgeted allocations. In *Proceedings of SODA '11: the 22nd ACM-SIAM Symposium on Discrete Algorithms*, pages 1253–1264. ACM-SIAM, 2011.

[4] H. Aziz, F. Brandt, and P. Harrenstein. Pareto optimality in coalition formation. *Games and Economic Behavior*, 82:562–581, 2013.

[5] H. Aziz, S. Gaspers, S. Mackenzie, and T. Walsh. Fair assignment of indivisible objects under ordinal preferences. *Artificial Intelligence*, 227:71–92, 2015.

[6] M. Babaioff, N. Immorlica, D. Kempe, and R. Kleinberg. Online auctions and generalized secretary problems. *SIGecom Exchanges*, 7(2), 2008.

[7] S. Bade. Random serial dictatorship: the one and only. *Mathematics of Operations Research,* to appear, 2018.

[8] A. Bhalgat, D. Chakrabarty, and S. Khanna. Social welfare in one-sided matching markets without money. In *Proceedings of APPROX+RANDOM '11: the 14th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems and the 15th International Workshop on Randomization and Computation*, volume 6845 of *Lecture Notes in Computer Science*, pages 87–98. Springer, 2011.

[9] A. Bogomolnaia and H. Moulin. A new solution to the random assignment problem. *Journal of Economic Theory*, 100(2):295 – 328, 2001.

[10] A. Bogomolnaia and H. Moulin. Random matching under dichotomous preferences. *Econometrica*, 72(1):257–279, 2004.

[11] A. Bogomolnaia and H. Moulin. Size versus fairness in the assignment problem. *Games and Economic Behavior*, 90:119–127, 2015.

[12] K. Cechlárová, P. Eirinakis, T. Fleiner, D. Magos, I. Mourtos, and E. Potpinková. Pareto optimality in many-to-many matching problems. *Discrete Optimization*, 14:160–169, 2014.

[13] D. Chakrabarty and C. Swamy. Welfare maximization and truthfulness in mechanism design with ordinal preferences. In *Proceedings of ITCS '14: the 5th Innovations in Theoretical Computer Science conference*, pages 105–120. ACM, 2014.

[14] N. Chen, N. Garvin, and P. Lu. Truthful generalized assignments via stable matching. *Mathematics of Operations Research*, 39(3):722–736, 2014.

[15] N. Chen and A. Ghosh. Algorithms for Pareto stable assignment. In V. Conitzer and J. Rothe, editors, *Proceedings of COMSOC '10: the 3rd International Workshop on Computational Social Choice*, pages 343–354. Düsseldorf University Press, 2010.

[16] N. R. Devanur, K. Jain, and R. Kleinberg. Randomized primal-dual analysis of RANKING for online bipartite matching. In *Proceedings of SODA '13: the 24th ACM-SIAM Symposium on Discrete Algorithms*, pages 101–107. ACM-SIAM, 2013.

[17] N.B. Dimitrov and C.G. Plaxton. Competitive weighted matching in transversal matroids. *Algorithmica*, 62(1-2):333–348, 2012.

[18] S. Dughmi and A. Ghosh. Truthful assignment without money. In *Proceedings of EC '10: the 11th ACM Conference on Electronic Commerce*, pages 325–334. ACM, 2010.

[19] P. Gärdenfors. Assignment problem based on ordinal preferences. *Management Science*, 20(3):331–340, 1973.

[20] A. Hylland and R. Zeckhauser. The efficient allocation of individuals to positions. *Journal of Political Economy*, 87(2):293–314, 1979.

[21] P. Jaramillo and V. Manjunath. The difference indifference makes in strategy-proof allocation of objects. *Journal of Economic Theory*, 147(5):1913–1946, 2012.

[22] R. M. Karp, U. V. Vazirani, and V. V. Vazirani. An optimal algorithm for on-line bipartite matching. In *Proceedings of STOC '90: the 22nd ACM Symposium on Theory of Computing*, pages 352–358. ACM, 1990.

[23] B. Korte and D. Hausmann. An analysis of the greedy heuristic for independence systems. In *Annals of Discrete Mathematics*, volume 2, pages 65–74. North-Holland, 1978.

[24] D.F. Manlove. *Algorithmics of Matching Under Preferences*. World Scientific, 2013.

[25] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[26] S. Pápai. Strategyproof assignment by hierarchical exchange. *Econometrica*, 68(6):1403–1433, 2000.

[27] C. G. Plaxton. A simple family of top trading cycles mechanisms for housing markets with indifferences. In *Proceedings of the 24th International Conference on Game Theory*, Stony Brook, New York, USA, 2013. Available from https://www.cs.utexas.edu/users/plaxton/pubs/2013/icgt.pdf.

[28] A.D. Procaccia and M. Tennenholtz. Approximate mechanism design without money. *ACM Transactions on Economics and Computation*, 1(4), 2013. Article number 18.

[29] A.E. Roth. Incentive compatibility in a market with indivisible goods. *Economics Letters*, 9:127–132, 1982.

[30] A.E. Roth and A. Postlewaite. Weak versus strong domination in a market with indivisible goods. *Journal of Mathematical Economics*, 4:131–137, 1977.

[31] A.E. Roth, T. Sonmez, and M.U. Ünver. Pairwise kidney exchange. *Journal of Economic Theory*, 125(2):151–188, December 2005.

[32] D. Saban and J. Sethuraman. House allocation with indifferences: A generalization and a unified view. In *Proceedings of EC '13: the 14th ACM Conference on Electronic Commerce*, pages 803–820. ACM, 2013.

[33] D. Saban and J. Sethuraman. The complexity of computing the random priority allocation matrix. *Mathematics of Operations Research*, 40(4):797–1088, 2015.

[34] L. Shapley and H. Scarf. On cores and indivisibility. *Journal of Mathematical Economics*, 1:23–37, 1974.

[35] L.-G. Svensson. Queue allocation of indivisible goods. *Social Choice and Welfare*, 11(4):323–330, 1994.

[36] T. Uno. Algorithms for enumerating all perfect, maximum and maximal matchings in bipartite graphs. In *Proceedings of ISAAC '97: the 8th International Symposium on Algorithms and Computation*, volume 1350, pages 92–101. Springer, 1997.

[37] A.C.-C. Yao. Probabilistic computations: Toward a unified measure of complexity (extended abstract). In *Proceedings of FOCS '77: the 18th Annual IEEE Symposium on Foundations of Computer Science*, pages 222–227. IEEE Computer Society, 1977.

[38] L. Zhou. On a conjecture by Gale about one-sided matching problems. *Journal of Economic Theory*, 52(1):123–135, 1990.

Appendix

In this section we prove Theorem 3.7 and also provide a systematic way of enumerating all Pareto optimal matchings. The following result is an important first step towards the former goal.

Theorem 3.8. *Any Pareto optimal matching is a strong priority matching for some ordering σ of the agents.*

Proof. This proof makes use of the characterisation of Pareto optimal matchings in instances of HA (potentially with ties) given by Proposition 2.1. Let a Pareto optimal matching μ be given.

Let $G = (V, E)$ be the *envy graph* for μ, defined as follows. In the graph, $V = N$ and there is a directed edge from agent i to agent k if and only if i weakly prefers $\mu(k)$ to $\mu(i)$. Every edge is colored. An edge (i, k) is colored green if $\mu(k) \simeq_i \mu(i)$, and is colored red otherwise—i.e., if $\mu(k) \succ_i \mu(i)$. We claim that all the edges in every strongly connected component (SCC) of G are green (we denote this claim by **C1**). To see this, note that by the definition of strongly connected components, there is a path from every node in a given component to every other node in the component. Hence, if there is a red edge (i, k) in a SCC, then there must be a cycle with a red edge in the SCC (as there must be a path from k to i). A cycle with at least one red edge corresponds to a cyclic coalition and hence μ could have not been a Pareto optimal matching, a contradiction.

Create graph $G' = (V', E')$ as follows. There is a vertex in V' for each SCC of G, and there is a directed edge in G' from v_r to v_s, $v_r, v_s \in V'$ and $v_r \neq v_s$, if and only if there is an edge in G from i to k for some i and k that belong to the SCCs of v_r and v_s. It follows from the definition of strongly connected components that G' is a DAG. Hence G' admits a topological ordering. Let X be a reversed topological ordering of G'. Let $\sigma = i_1, \ldots, i_{n_1}$ be an ordering of all the agents that is consistent with X. That is, for every two agents i_j and i_r, $1 \leq j < r \leq n_1$, the corresponding SCC of i_j appears in X no later than the corresponding SCC of i_r. (The order of the agents belonging to the same SCC can be determined arbitrarily.) We prove that μ is an SPM w.r.t. σ.

Assume, for a contradiction, that our claim does not hold. That is, μ is not an SPM w.r.t. σ. Hence there must exist another matching μ' which has a lexicographically smaller signature than μ; i.e., $\rho(\mu') < \rho(\mu)$ (we denote this fact by **A1**). Let i_j be the highest priority agent, w.r.t σ, such that i_j strictly prefers his partner under μ' to his partner under μ; i.e., $\mu'(i_j) \succ_{i_j} \mu(i_j)$ (we denote this assumption by **A2**). Note that $\mu'(i_j)$ must be matched in μ or else μ admits an alternating coalition, namely $P = \langle i_j, \mu'(i_j) \rangle$—as i_j strictly prefers unmatched object $\mu'(i_j)$ to his partner $\mu(i_j)$—and hence not Pareto optimal. So $\mu'(i_j)$ is matched under μ to, say, i_k. Following **A2**, there must be a red edge from i_j to i_k in the envy graph G. Therefore, i_k must have a higher priority than i_j according to σ (note that, by **C1**, i_j and i_k cannot belong to the same SCC). It then follows from **A1** and **A2** that i_k is matched under μ' and ranks his partners under μ and μ' the same; i.e. $\mu(i_k) \simeq_{i_k} \mu'(i_k)$. Now, $\mu'(i_k)$ must be matched in μ or else there is an alternating path coalition in μ, namely $P = \langle i_j, i_k, \mu'(i_k) \rangle$, and hence μ is not Pareto optimal. Also, i_k cannot be matched to $\mu(i_j)$ or else there is a cyclic coalition in μ, namely $P = \langle i_j, i_k \rangle$, and hence μ is not Pareto optimal. So $\mu'(i_k)$ is matched under μ to, say, i_r, $i_r \neq i_j$. Also, there is a green edge from i_k to i_r in G, since $\mu'(i_k) = \mu(i_r)$ and $\mu(i_k) \simeq_{i_k} \mu'(i_k)$. We claim that i_r must have a higher priority than i_j (we denote this claim by **C2**). To see this, first of all note that there cannot exist a path between i_r and i_j in G or else G admits a cycle with at least one red edge, namely (i_j, i_k), and hence μ admits a cyclic coalition and is not Pareto optimal. Now, to complete the proof of **C2**, we consider two cases regarding whether there is a path from i_r to i_k or not.

- Case 1: *There is a path from i_r to i_k in G.* Since there is an path from i_k to i_r, then i_r and i_k must belong to the same strongly connected component. Therefore since i_k has a higher priority than i_j under σ, so must i_r.

- Case 2: *There is no path from i_r to i_k in G.* Therefore i_r and i_k belong to two different SCCs. Since there is an path from i_k to i_r in G, there is an edge in G' from the SCC corresponding to i_k to the SCC corresponding to i_r. Therefore i_r must have a higher priority than i_k under X and thus under σ as well. Hence, by transitivity, i_r has a higher priority than i_j under σ.

So far we have established that i_r has a higher priority than i_j. Now, using a similar argument as for i_k, we can show that i_r must be matched under μ' and must rank his partners under μ and μ' the same. Also, again using a similar argument as for i_k, $\mu'(i_r)$ is not the same object as $\mu(i_j)$ and $\mu'(i_r)$ must be matched in μ or else there is an alternating path coalition in μ contradicting the Pareto optimality of μ. So $\mu'(i_r)$ is matched under μ, to say i_q. Using exactly the same argument as we used for i_r we can show that i_q also has a higher priority than i_j. We can keep repeating this argument and every time we have to reach a new agent with a higher priority than i_j. However, there are a bounded number of agents and hence a bounded number of agents with higher priority than i_j, a contradiction. $\square$

Using Theorem 3.8, we show that SDMT-1 is capable of producing any given Pareto optimal matching.

Theorem 3.7. *Any Pareto optimal matching can be generated by some execution of SDMT-1.*

Proof. Let μ be a Pareto optimal matching for an instance I of HA. By Theorem 3.8, μ is an SPM for some ordering σ of the agents. Execute SDMT-1 given σ as follows. At each phase i, choose $(i, \mu(i))$ as the augmenting path. Notice that since μ is a matching, both i and $\mu(i)$ must be unmatched at the beginning of phase i. Futhermore, since μ is an SPM w.r.t σ, there cannot be an augmenting path from i to an object that i strictly prefers to $\mu(i)$. $\square$

The above theorem implies that any Pareto optimal matching has a nonzero chance of materialising if we execute the following procedure: (1) randomly generate a priority ordering over the agents σ, (2) run SDMT-1 given σ, and (3) whenever faced with more than one choice, pick an augmenting path at random. Enumerating all Pareto optimal matchings in a more systematic way is however possible with the aid of Theorem 3.8 and the two forthcoming propositions, Proposition 3.9 and Proposition 3.10.

Following Theorem 3.8, enumerating all Pareto optimal matchings is equivalent to enumerating all matchings μ such that μ is a strong priority matching w.r.t. some ordering of the agents. Recall that all SPMs w.r.t. σ have the same signature. It hence follows that:

Proposition 3.9. *Let μ^* be a matching returned by SDMT-1 for a given priority ordering of the agents $\sigma = i_1, \ldots, i_{n_1}$. Then, a given matching μ is a strong priority matching w.r.t. σ if and only if*

- *the same set of agents are matched under both μ and μ^*, and*

- *each matched agent i is matched under μ to an object that he ranks the same as $\mu^*(i)$; i.e., $rank(i, \mu(i)) = rank(i, \mu^*(i))$.*

Let $G(\sigma) = (V, E)$ be a graph where $V = N^* \cup O$ where N^* is the set of agents that are matched under μ^*. There is an edge between an agent $i \in N^*$ and an object $o \in O$ if and only if i ranks o the same as $\mu^*(i)$. It is then easy to see that:

Proposition 3.10. *A matching μ is a strong priority matching w.r.t. σ if and only if it is a maximum cardinality matching in $G(\sigma)$.*

Theorem 3.11. *Given and instance I of HA, all Pareto optimal matchigns of I can be enumerated in time $O(n_1!)$.*

Proof. It follows Theorem 3.8 that to enumerate all Pareto optimal matchings it is enough to execute the following procedure on all possible σ: (1) run SDMT-1 given σ, and (2) enumerate all SPMs w.r.t. σ. It follows Proposition 3.10 that to enumerate all SPMs w.r.t. σ we need only to enumerate all maximum cardinality matchings w.r.t. $G(\sigma)$. The latter can be achieved in $O(|V|)$ time per matching [36]. $\square$

This figure "tradinggraph.png" is available in "png" format from:

http://arxiv.org/ps/1404.5245v6