Algorithms for Pareto Stable Assignment

Ning Chen and Arpita Ghosh

Abstract

Motivated by online matching marketplaces, we study stability in a many-to-many market with ties and incomplete preference lists. When preference lists contain ties, stable matchings need not be Pareto optimal. We consider the algorithmic question of computing outcomes that are both Pareto optimal and stable in a many-to-many two-sided market with ties and incomplete lists, where agents on *both* sides can have multi-unit capacities, as well as trade multiple units with the same neighbor.
Our main result is a fast algorithm for computing Pareto-stable assignments for this very general multi-unit matching problem with arbitrary preference lists on both sides, with running time that is polynomial in the *number of agents* in the market, rather than the sum of capacities of all agents.

1 Introduction

A fundamental solution concept in the context of two-sided matching marketplaces is that of *stability*, introduced by Gale and Shapley in their seminal work on stable marriage [10]. In the marriage model, there are n men and n women, each with a strict preference ranking over all members of the other side: a matching between the men and women is *stable* if there is no unmatched man-woman pair who both prefer each other to their current partners. The concept of stability has had enormous influence both on the design of real world matching markets [25] as well as its theory [27] — a number of variants of the stable matching problem have been studied, relaxing or generalizing different assumptions in the original model.

One particularly practical generalization is to relax the requirement of strict and complete preferences over all alternatives to accommodate indifferences and intolerance — a man can have an *incomplete* preference list, *i.e.*, he need not rank all women, and can have *ties*, *i.e.*, he can be indifferent between some women in his preference list (and similarly for women). The introduction of ties and incomplete lists dramatically changes the properties and structure of the set of stable matchings relative to the Gale-Shapley marriage model, and often leads to interesting algorithmic and computational questions in the context of choosing amongst the large set of stable matchings. For instance, man or woman-optimal stable matchings[1] are no longer well-defined [27]; stable matchings need not all have the same cardinality, and the problem of finding the maximum cardinality stable matching becomes NP-hard [16].

One of the most important differences that arises due to indifferences in preference lists, however, is that *stability no longer guarantees Pareto optimality*[2], an observation that has received a great deal of attention in the economics literature (see, for example, [2, 7, 1, 29, 8, 9]). When preference lists contain ties, not all stable matchings are Pareto optimal and in fact, as [8] demonstrates, simply using a matching returned by the Gale-Shapley deferred acceptance algorithm can cause quite a severe loss in efficiency. A natural question, then, is whether one can find a matching which is both stable and Pareto optimal when preferences may contain ties. This question has recently been addressed for the *many-to-one* matching

[1] The outcome of the man-proposing deferred acceptance algorithm with strict complete preferences is a man-optimal stable matching.

[2] A simple example consists of two men and two women, where i_1 strictly prefers j_1 to j_2, but all other nodes are indifferent amongst their possible partners. The matching $(i_1, j_2), (i_2, j_1)$ is stable, but not Pareto optimal since i_1 can be reassigned to j_1 and i_2 to j_2 without making anyone worse off.

model — Erdil and Ergin [8, 9] give an algorithm that finds a Pareto-stable many-to-one matching, with runtime that is polynomial in the total capacity of all agents.

In this paper, we study the problem of efficiently computing a Pareto-stable outcome in a very general *many-to-many* setting with indifferences: agents on *both* sides can have multi-unit capacities, as well as trade multiple units with the same neighbor. (Observe that here, unlike the many-to-one setting, the total capacity of nodes in the graph is not restricted to be polynomial in the size of the graph.) The many-to-many setting has attracted growing interest in the economics literature both because of an increasing number of applications (indeed, a number of online marketplaces are many-to-many since both buyers and sellers have multi-unit demand and supply), and more importantly because of fundamental theoretical differences from the well understood many-to-one setting. We focus here on a *computational* problem that arises from allowing nodes on both sides to have multi-unit capacities— while a naive adaptation of the many-to-one algorithm would return a Pareto-stable assignment, it would do so in time that grows polynomially with the total capacity of all nodes in the graph, rather than the size of the graph itself. We therefore seek a strongly polynomial time algorithm for the problem of computing a Pareto optimal stable assignment.

The computer science literature on algorithms for stable matching in the presence of ties and incomplete lists has largely focused on the problems of deciding the existence of, and computing, strongly-stable and super-stable matchings [15], and computing stable matchings with large size or weight. While the problem of finding a strongly stable matching if it exists can be solved in polynomial time [15, 17] and the resulting outcomes are indeed Pareto optimal, such matchings need not always exist, making them an unsuitable solution concept practically. Also, the problem of finding the maximum cardinality stable matching is NP-hard [16]. The solution concept of Pareto stability offers a strict refinement of the set of stable matchings, and in addition, has the important property that it always exists, and, as we show, can be computed efficiently. Given that choosing a globally optimal stable matching is difficult, Pareto-stable matchings, which are *locally optimal*, are a natural choice amongst stable matchings — a stable matching which is not Pareto optimal unnecessarily compromises efficiency, since it is possible to make some agents strictly better off without compromising the welfare of any other agents.

1.1 Our Results

Our main result is an algorithm that finds a Pareto-optimal stable assignment, with running time that is polynomial in the *number of nodes* in the graph, for a many-to-many two-sided market where: (i) all nodes can have ties and incomplete preference lists over the other side, (ii) nodes on both sides have multi-unit capacities, (iii) there can be multiple edges between a pair (i, j) (*i.e.*, multiple units can be assigned between i and j). While ties and incomplete lists motivate Pareto stability, the actual technical challenge arises due to the multi-unit node capacities, which, unlike in the many-to-one setting of [8, 9], need not be polynomial in the size of the graph.

With unit capacity (matching), a fairly straightforward application of standard notions of augmenting paths and cycles from network flows [18] leads to an algorithm that finds Pareto-optimal stable matchings [29]. A naive approach to the many-to-many matching problem is to simply make identical copies of nodes, one for each unit of its capacity, and compute a Pareto-stable matching for this equivalent instance, using the algorithm for the relatively simple unit supply/demand setting. However, the size of this instance is proportional to the total capacity of all nodes, and therefore will not give us a strongly polynomial time algorithm. Instead, we construct a sequence of modified networks with one copy of a node for *each level in its preference list* (the number of levels in a preference list cannot exceed the number of nodes)— this allows us to correctly define the notion of

"improvement edges" (§4.1) when nodes have multiunit capacities. The second challenge is to ensure that once all Pareto-improvements at a certain preference level for a particular node have been found, the reassignments made by the algorithm for a *different* node or level does not *reintroduce* Pareto-improvements for this node and preference level (Example 4.1 demonstrates that this can indeed happen for a only slightly different (and perhaps more natural) network construction). We use maximum flow computations on a series of carefully designed augmented networks such that increases in flow preserve stability and correspond to Pareto improvements in the assignment, and there are no remaining Pareto improvements after all networks have been executed once. The algorithm and its proof of correctness are given in Section 4.

Applications. The many-to-many matching problem has recently attracted growing interest because of a number of applications such as job markets where applicants seek multiple part-time positions [6], auto markets [12], as well as electronic marketplaces such as eBay, and online advertising exchanges. A specific application in the electronic marketplace setting is in the context of *social lending* [4], which is a large and rapidly expanding marketplace for matching lenders and borrowers directly without the use of traditional financial intermediaries. In the social lending marketplace, lenders have preferences over borrowers since they each represent investments with different *risk* levels— so a lender might prefer to invest in some borrowers more than others, even amongst the set of acceptable borrowers. While lenders have explicit preferences over borrowers, the interest rates offered by lenders can be used to define a preference ranking over lenders for the borrower side of the graph as well. The question of how to clear this two-sided matching market leads immediately to our problem of efficiently computing Pareto-stable assignments, since both lenders and borrowers have multiunit capacities (lending budgets and desired loan amounts respectively [4]), with preferences that are incomplete and contain ties[3]. The need for computational efficiency is particularly striking in this setting, since an algorithm that is polynomial in the total capacity of the instance, *i.e.*, the total volume of loans in the market, as opposed to the total number of agents (lenders and borrowers) is clearly not efficient.

The social lending site Zopa, with over 400,000 members and $50 million in loans, already uses a centralized matching system where lenders can specify bids for each category (arranged by credit-rating) of borrowers and a total budget, but not preferences across categories[4]. Our algorithm would permit offering a more expressive bidding language for lenders, which allows specifying preferences across categories in addition to the total budget and bids, by providing a solution for the market-clearing problem.

1.2 Related Work

Two-sided matchings have been studied extensively since the seminal work of Gale and Shapley on stable marriage [10]. There is now a vast literature studying various aspects of the original stable marriage model as well as many of its variants, such as ties in preference lists, incomplete preferences, and weighted edges, as well as non-bipartite versions such as the roommate model. For a nice review of the very large economics literature on the subject, see the book by Roth and Sotomayor [27] and the survey by Roth [25]; for an introduction to the computer science literature addressing algorithmic and computational questions, see, for instance, [11, 3, 14].

[3]Ties are ubiquitous in social lending, since lenders can often only distinguish between borrowers by credit-rating. Preference lists can be incomplete since some borrowers, for instance those with poor credit rating, may not be acceptable to a lender.

[4]A lender can specify separate budgets for each category, but clearly this is a strict subset of the expressiveness offered by allowing budgets along with preferences over categories

The papers most relevant to our work from the stable matching literature are the following. Sotomayor [29] proposes Pareto-stable matchings as a natural solution concept for a many-to-many marketplace and studies structural aspects of Pareto-stable matchings. As previously discussed, Erdil and Ergin [8, 9] study the algorithmic question of finding Pareto-optimal matchings for the *many-to-one* setting and give an algorithm whose running time is polynomial in the sum of capacities of all nodes in the graph.

The many-to-many stable matching problem is far less well-studied, with a small, but growing, body of research, motivated by practical settings such as electronic marketplaces, and job markets where some agents might seek multiple part-time positions [6]. The generalization to multi-unit node capacities on both sides is nontrivial: as Echenique and Oviedo [6] show, even a small number of agents with multi-unit capacity drastically alter the properties of matchings compared to the many-to-one setting. Much of the literature on many-to-many stable matchings focuses on settings without indifferences: Hatfield and Kominers [12] study stability in very general model with bilateral contracts and prove necessary conditions for the existence of stable matchings as well as results regarding the structure of the set of stable matchings. Echenique and Oviedo [6] show the equivalence of different solution concepts under strong substitutability for many-to-many matching, also in a setting with strict preferences. Finally, Malhotra [19] studies the algorithmic question of finding *strongly stable* matchings, if they exist, in a many-to-many matching model with ties and complete lists.

2 Model

There is an underlying bipartite graph M with nodes, or agents, (A, B) and edge set E. The existence of an edge (i, j) means agents $i \in A$ and $j \in B$ are mutually willing to be matched with, or assigned to, each other.

Each node in M has a *capacity constraint*, which is the maximum number of units that it can trade with its neighbors: we denote by this capacity by c_i. We will assume that the capacities c_i are integers, that is, the capacities are discrete rather than continuous (this assumption is easily justifiable for the natural applications of stable assignment). The presence of node capacities allows us to assume, without loss of generality, that $|A| = |B| = n$, since dummy nodes with $c_i = 0$ can be added to the market to ensure that there is an equal number of nodes on both sides.

We use the term *assignment* as a generalization of matching to our many-to-many setting to mean a multi-unit pairing between the nodes in A and B. A *feasible assignment* $X = (x_{ij})_{(i,j)\in E}$, where $x_{ij} \geq 0$ is the number of units assigned between $i \in A$ and $j \in B$, satisfies capacity constraints on both sides, that is, $\sum_j x_{ij} \leq c_i$ and $\sum_i x_{ij} \leq c_j$. Note that both inequalities can be strict in a feasible assignment, that is, a node's capacity need not be exhausted completely. When all nodes have unit capacity, a feasible assignment is identical to a bipartite *matching*.

Preference Model. Each node $i \in A$ (respectively $j \in B$) has a preference list P_i ranking its neighbors $\{j \in B : (i, j) \in E\}$ (respectively $\{i \in A : (i, j) \in E\}$). The preference lists are allowed to have *ties*, *i.e.*, a node can be indifferent amongst any subset of its neighbors. Since a node's preference list is restricted to the set of its neighbors, the preference list is naturally incomplete. For example, a possible preference list for node i is $P_i = ([j_1, j_2], [j_3, j_5])$: that is, i is indifferent between j_1 and j_2, and prefers either of them to j_3, j_5, which i is indifferent amongst, and finds all other partners unacceptable.

Definition 2.1 (Level function)**.** *We use the function $L_i(\cdot)$ to encode the preference list of a node $i \in A$ over individual nodes in B: for each $j \in P_i$, $L_i(j) \in \{1, \ldots, n\}$ gives the ranking of j in i's preference list. That is, for any $j, j' \in P_i$, if $L_i(j) < L_i(j')$, then i strictly*

prefers j to j'; if $L_i(j) = L_i(j')$, then i is indifferent between j and j'. (In the example above, $L_i(j_1) = L_i(j_2) = 1$ and $L_i(j_3) = L_i(j_4) = L_i(j_5) = 2$.)

The definition of the level function $L_j(\cdot)$ for each $j \in B$ is symmetric.

The preferences of nodes over individual neighbors define a natural ranking over sets of neighbors, which we use to define the preference of a node over sets of neighbors: Given sets of neighbors S and S', arrange the nodes in S and S' in decreasing order of rank in i's preference list. i prefers S to S' if and only if $j_l \succeq j'_l$ for each l (using $\emptyset$ to make the sets equal-sized if one set has fewer neighbors than the other). Note that this is only a partial order, and specifically, some sets may not be comparable— for example, i cannot compare (or equivalently, is indifferent between) the sets $\{j_1, j_4\}$ and $\{j_2, j_3\}$, where j_l is at level l in i's preference list. This model of preferences for nodes with multi-unit capacity is both natural and has the advantage that nodes continue to only express preferences over individuals, and is exactly that used by Erdil and Ergin [8, 9] in their work on Pareto-stability for many-to-one matchings. We note that the choice of preference model over *sets* is relevant only to the Pareto optimality component of our discussion, and does *not* matter for stability, which is a pairwise solution concept and is not affected by preferences over sets.

3 Pareto-Stability

We first state the definition of stability for assignment; again, we use the term *stable assignment* to make the distinction with the unit-capacity setting, where an assignment reduces to a matching.

Definition 3.1 (Stable assignment)**.** *We say that an assignment $X = (x_{ij})$ is* stable *if there is no blocking pair (i, j), $i \in A$ and $j \in B$, $(i, j) \in E$, satisfying the one of the following conditions:*

- *Both i and j have leftover capacity;*
- *i has leftover capacity and there is i', $x_{i'j} > 0$, such that j strictly prefers i to i'; or j has capacity remaining and there is j', $x_{ij'} > 0$, such that i prefers j to j';*
- *There are i' and j', $x_{ij'} > 0$ and $x_{i'j} > 0$, such that i strictly prefers j to j' and j strictly prefers i to i'.*

Note that both members of a blocking pair must strictly prefer each other to their current partners. A stable assignment always exists, and can be found using a variant of Gale-Shapley algorithm [10] for computing stable matchings. We next define Pareto optimal assignments.

Definition 3.2 (Pareto-optimal assignment)**.** *Given assignment $X = (x_{ij})$, let $x_i(\alpha) = \sum_{j:\ L_i(j) \leq \alpha} x_{ij}$ be the total number of units of i's capacity that is assigned at levels better than or equal to level α, and $x_j(\beta) = \sum_{i:\ L_j(i) \leq \beta} x_{ij}$ be the total number of assigned units of j's capacity that are better than or equal to level β. We say that $X = (x_{ij})$ is* Pareto-optimal *if there is no other feasible assignment $Y = (y_{ij})$ such that $y_i(\alpha) \geq x_i(\alpha)$ and $y_j(\beta) \geq x_j(\beta)$, for all α, β, and at least one of the inequalities is strict.*

Recall from §1 that when preference lists contain ties, a stable matching need not be Pareto optimal. This leads naturally to the concept of Pareto stable matchings [29], which combines both Pareto-optimality and stability to provide a stronger solution concept to choose from amongst the set of stable matchings. (Note that the presence of ties in preference lists cannot be addressed by the standard trick of breaking ties using small perturbations:

if ties are broken arbitrarily, the set of stable matchings with respect to the new strict preferences can be strictly smaller than the set of stable matchings with respect to the original preferences with ties— that is, artificial tiebreaking does not preserve the set of stable matchings in the original problem.)

Definition 3.3. *A* Pareto-stable *assignment is a feasible assignment that is both stable and Pareto optimal.*

Augmenting Paths and Cycles. Given the connection between assignment and network flow, it is not surprising that the existence of augmenting paths and cycles in an assignment is closely related to whether it can be improved, *i.e.*, its Pareto optimality. The main difference in the context of stable matching is that nodes have preferences in addition to capacities: thus, augmenting paths and cycles must improve not just the size of an assignment, but also its quality, as determined by node preferences. We first define augmenting paths and cycles in the context of stable assignment.

Definition 3.4 (Augmenting Path)**.** *Given an assignment $X = (x_{ij})$, we say that $[i_0, j_1, i_1, \ldots, j_\ell, i_\ell, j_{\ell+1}]$ is an* augmenting path *if (i) $x_{i_0} < c_{i_0}$ and $x_{j_{\ell+1}} < c_{j_{\ell+1}}$, (ii) $x_{i_k j_k} > 0$ and $x_{i_{k-1} j_k} < c_{i_{k-1} j_k}$ for $k = 1, \ldots, \ell$, and (iii) $L_{i_k}(j_k) \geq L_{i_k}(j_{k+1})$ and $L_{j_k}(i_{k-1}) \leq L_{j_k}(i_k)$ for $k = 1, \ldots, \ell$.*

Definition 3.5 (Augmenting Cycle)**.** *Given an assignment $X = (x_{ij})$, we say that $[i_1, j_2, i_2, \ldots, j_\ell, i_\ell, j_1, i_1]$ is an* augmenting cycle *if (i) $x_{i_k j_k} > 0$ and $x_{i_{k-1} j_k} < c_{i_{k-1} j_k}$ for $k = 1, \ldots, \ell$, (ii) $L_{i_k}(j_k) \geq L_{i_k}(j_{k+1})$ and $L_{j_k}(i_{k-1}) \leq L_{j_k}(i_k)$ for $k = 1, \ldots, \ell$, where $i_0 = i_\ell$ and $j_{\ell+1} = j_1$, and (iii) at least one of the above inequalities is strict. If i_k is such a node (resp. j_k), we say it is an augmenting cycle associated with i_k (resp. j_k) at level $L_{i_k}(j_k)$ (resp. $L_{j_k}(i_k)$).*

The following easy lemma implies that if a stable assignment has no augmenting paths or cycles, then it must be Pareto stable (a similar result for Pareto-stable *matching* was shown in [29].)

Lemma 3.1. *Any assignment X that has no augmenting path or cycle is Pareto-optimal.*

4 Computing a Pareto-Stable Assignment

We now give a strongly polynomial time algorithm to compute a Pareto-stable assignment. Note that if X is a stable assignment, reassigning according to any augmenting path or cycle of X preserves stability, *i.e.*, any assignment Y that Pareto dominates a stable assignment X is stable as well [9]. This, together with Lemma 3.1, suggests that starting with a stable assignment, and then making improvements to it using augmenting paths and cycles until no more improvements are possible, will result in a Pareto stable assignment.

How do we find such augmenting paths and cycles? First consider the simplest case with unit capacity, *i.e.*, $c_i = c_j = 1$ for all i, j; here, an assignment degenerates to a matching. Given an existing matching, define a new directed bipartite graph with the same nodes, where all forward edges are *"weak improvement"* edges with respect to the existing matching, and backward edges correspond to the pairings in current matching. Then we are able to find augmenting paths by introducing a source and sink that link to unmatched nodes on each side. For cycles, since we need strict improvement for at least one node, we consider subgraphs, one for each node, which only consists of strict improvement edges for that node; then any cycle in the subgraph containing that node gives an augmenting cycle.

For our general case where $c_i \geq 1$, however, note that *even the concept of improvement edges for a node is not well defined* — since a node can have multiple partners in an assignment, a particular edge can present an improvement for some part of that node's capacity and not for some others. For instance, suppose that node i (with $c_i = 2$) is matched to nodes j_1 and j_3, and suppose that i strictly prefers j_1 to j_2 to j_3. Then, (i, j_2) would only represent an improvement relative to (i, j_3), but not with respect to (i, j_1), both of which exist in the current assignment.

An obvious way to fix this problem is to make copies of each node, one for each unit of its capacity, in which case improvement edges are well-defined– each unit of flow is associated with a unique neighbor in any assignment. However, note that this new graph has size $\sum_i c_i + \sum_j c_j$, consequently computing a Pareto-stable assignment in time polynomial in $\sum_i c_i + \sum_j c_j$, which, alas, is exponential in the size of the input.

4.1 Construction of Networks

In order to define improvement edges in this setting with multiple units of supply and demand, we will create a new augmented bipartite graph from the original bipartite graph and preference lists of nodes. The vertex set consists of copies of each node, where each copy represents a level on that node's preference list. We then define forward and backward edges between the vertices: forward edges are the (weak) improvement edges, while there is one backward edge for every feasible pair (i, j), $i \in A, j \in B$, corresponding to their respective levels in the others' preference list. This augmented graph, which is assignment-independent and depends only on the preference lists of the nodes, is then used to define a sequence of networks with assignment-*dependent* capacities, which allow us to find augmenting paths and cycles. The constructions are described formally below.

Definition 4.1. *Given the preference lists of nodes, we construct a directed graph G as follows.*

- *Vertices: For each node $i \in M$ (either in A or B), we introduce n new vertices $T(i) = \{i(1), \ldots, i(n)\}$, where $i(\alpha)$ corresponds the α-th level of the preference list of i. (If i has $k < n$ levels in his preference list, it suffices to introduce only k vertices $i(1), \ldots, i(k)$; here, we use n levels for uniformity.)*
- *Edges: For each pair $(i, j) \in E(M)$, let $\alpha = L_i(j)$ and $\beta = L_j(i)$. We add a* backward *edge between $i(\alpha)$ and $j(\beta)$, i.e. $j(\beta) \to i(\alpha)$. Further, we add a* forward *edge $i(\alpha') \to j(\beta')$ for every pair of vertices $i(\alpha')$ and $j(\beta')$ satisfying $\alpha' \geq \alpha$ and $\beta' \geq \beta$.*

The following figure shows an example of the construction, where the first figure gives the input instance (the number on each node is its supply/demand).

Definition 4.2 (Network H)**.** *Given graph G and an assignment X, we define network $H(X)$ as follows. We assign capacity infinity to all forward edges in G, and capacity x_{ij} to the backward edge between $T(i)$ and $T(j)$. We include a source s and a sink t; further, for each $i \in A$ and $j \in B$, we add an extra vertex h_i and h_j, respectively. We connect $s \to h_i$ with capacity $c_i - x_i$, and $h_j \to t$ with capacity $c_j - x_j$, where $x_i = \sum_j x_{ij}$ and $x_j = \sum_i x_{ij}$. Further, we connect $h_i \to i(\alpha)$ with capacity infinity for $\alpha = 1, \ldots, n$, and connect $j(\beta) \to h_j$ with capacity infinity for $\beta = 1, \ldots, n$.*

We will use the network H to find augmenting paths with respect to an existing stable assignment X. Observe that the only edges from the source with nonzero capacity are those that connect to a node $i \in A$ with leftover capacity; similarly, the only edges to the sink with nonzero capacity are from a node $j \in B$ with leftover capacity. Sending flow from s to t in

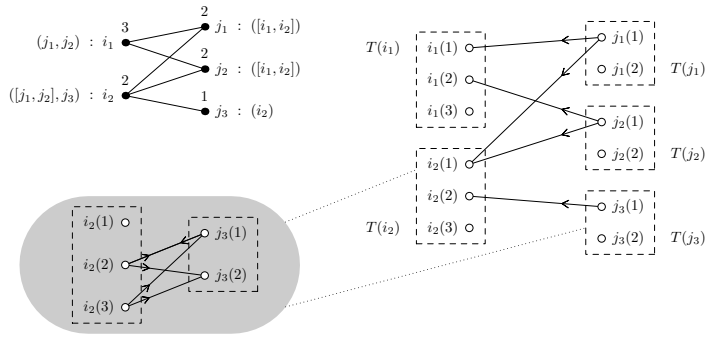

Figure 1: Construction of graph G.

H therefore involves increasing the total size of the assignment, exactly as in an augmenting path for X. In fact, as we will show in Proposition 4.1, after finding the maximum flow in H and updating the assignment accordingly, there are no remaining augmenting paths in the new assignment.

Definition 4.3 (Networks $H_{i,\alpha}$ and $H_{j,\beta}$)**.** *Given the graph G and X, let $G(X)$ be the network where all forward edges in G are assigned capacity infinity, and all backward edges are assigned capacity x_{ij}. We use $G(X)$ define the networks $H_{i,\alpha}(X)$ and $H_{j,\beta}(X)$ for each $i \in A$ and $j \in B$, and $\alpha, \beta = 1, \ldots, n$, as follows.*

To get network $H_{i,\alpha}(X)$ from $G(X)$, we add a source s and a sink t, and connect $s \to j(\beta)$ with capacity infinity for each vertex $j(\beta)$ satisfying $\alpha > L_i(j)$ and $\beta \geq L_j(i)$ (an equivalent definition is that we connect $s \to j(\beta)$ if there is an edge $i(\alpha) \to j(\beta)$ and $\alpha > L_i(j)$). Further, we connect $j(\beta) \to t$ with capacity x_{ij} for each $j(\beta)$ satisfying $\alpha \leq L_i(j)$ and $\beta = L_j(i)$.

The network $H_{j,\beta}(X)$ is defined symmetrically. That is, we include a source s and a sink t, and connect $s \to i(\alpha)$ with capacity x_{ij} for each vertex $i(\alpha)$ satisfying $\alpha = L_i(j)$ and $\beta \leq L_j(i)$. Further, we connect $i(\alpha) \to t$ with capacity infinity for each $i(\alpha)$ satisfying $\alpha \geq L_i(j)$ and $\beta > L_j(i)$.

We will use the networks $H_{i,\alpha}$ and $H_{j,\beta}$ to find augmenting cycles associated with i and j at level α and β, respectively. Consider any flow from s to t in $H_{i,\alpha}$, say $[s, j_1(\beta_1), i_1(\alpha_1), \ldots, i_2(\alpha_2), j_2(\beta_2), t]$, we know that $\alpha > L_i(j_1)$ (i.e. i strictly prefers j_1 to all its neighbors at level α) and $L_{j_1}(i_1) = \beta_1 \geq L_{j_1}(i)$ (i.e. j_1 weakly prefers i to i_1). Further, we have $\alpha \leq L_i(j_2)$ (this implies that i strictly prefers j_1 to j_2) and $L_{j_2}(i_2) \leq \beta_2 = L_{j_2}(i)$ (i.e. j_2 weakly prefers i_2 to i). That is, flows from s to t in $H_{i,\alpha}$ correspond to augmenting cycles for node i at levels less than or equal to α in X (a symmetric argument holds for graph $H_{j,\beta}$). We will show in Proposition 4.2 that once we compute the maximum flow in $H_{i,\alpha}$, there are no remaining augmenting cycles for node i at level α (note, not level α or below).

4.2 Algorithm

PARETO STABLE ASSIGNMENT (PARETO-ASSIGNMENT)

1. Let X be an arbitrary stable assignment
2. Construct networks H, $H_{i,\alpha}$ and $H_{j,\beta}$, for each $i \in A$, $j \in B$, and $\alpha, \beta = 1, \ldots, n$ given X
3. For H, $H_{i,\alpha}$ and $H_{j,\beta}$ constructed above (H to be executed first)
   (a) Compute a maximum flow $F = (f_{uv})$ from s to t (if there is no flow from vertex u to v, set $f_{uv} = 0$)
   (b) For each forward edge $i(\alpha) \to j(\beta)$, let $x_{ij} = x_{ij} + f_{i(\alpha)j(\beta)}$
   (c) For each backward edge $j(\beta) \to i(\alpha)$, let $x_{ij} = x_{ij} - f_{j(\beta)i(\alpha)}$
   (d) If the graph is $H_{i,\alpha}$
      - Let $x_{ij} = x_{ij} + f_{sj(\beta)}$ for each edge $s \to j(\beta)$
      - Let $x_{ij} = x_{ij} - f_{j(\beta)t}$ for each edge $j(\beta) \to t$
   (e) If the graph is $H_{j,\beta}$
      - Let $x_{ij} = x_{ij} - f_{sj(\beta)}$ for each edge $s \to i(\alpha)$
      - Let $x_{ij} = x_{ij} + f_{j(\beta)t}$ for each edge $i(\alpha) \to t$
   (f) Update the capacities for the next graph to be executed according to the new assignment X
4. Output X (denoted by X^*)

To prove that PARETO-ASSIGNMENT indeed computes a Pareto-stable assignment, we need to show two main things — first, that the resulting assignment is feasible, stable, and all nodes' assignments are weakly enhanced through the course of the algorithm.

Second, we need to show that no further Pareto improvements are possible when the algorithm terminates, *i.e.*, X^* is Pareto optimal. Note that the assignment X changes through the course of the algorithm, and therefore we need to show that, for instance, no other augmenting paths can be found after the network H has been executed, *even though* the assignment X that was used to define the network H has been changed (and similarly for all augmenting cycles). That is, while we compute maximum flows in $H(X)$ to find all augmenting paths for a given assignment X, we need to show that no new augmenting paths have showed up in the updated assignments X' computed by the algorithm. Similarly, finding (i, α) augmenting cycles via $H_{i,\alpha}(X)$ for *some* assignment X does not automatically imply that no further (i, α) augmenting cycles will ever be found in *any* of the (different) assignments X' computed through the course of the algorithm, since the assignments of all nodes can change each time when a maximum flow is computed, leading to the possibility of new valid s-t paths, and therefore possibly new augmenting cycles. Note that this is hardly obvious, and in fact, as Example 4.1 demonstrates, that this does not happen is due to a careful choice of the construction of the networks $H_{i,\alpha}, H_{j,\beta}$.

Example 4.1. *Suppose there are four nodes i_1, i_2, i_3, k in A and five nodes j_1, j_2, j_3, j_4, j_5 in B. All nodes except k have unit capacity and are indifferent between all possible partners (i.e., have only one level in their preference list). Node k has capacity 2, and preference list $([j_1, j_5], [j_3, j_4], j_2)$. Suppose we start with the (stable) assignment X_0 where k is matched to j_2, j_3, and the remaining assignments are $(i_1, j_1), (i_2, j_4), (i_3, j_5)$ (note there are no augmenting paths in X_0). Consider finding the maximum flow in network $H_{i,\alpha}$* without *the link $j_2 \to t$ for $\alpha = 2$. In this network, the total capacity of edges incident to the sink is 1, thus we can send at most one unit flow, for example $k \to j_1 \to i_1 \to j_2 \to k \to j_4 \to i_2 \to j_3 \to t$*

(note that the two k's here correspond to different vertices in $T(k)$ in the network). After reassigning assignment according to this flow, we obtain the new assignment X' $(i_1, j_2), (i_2, j_3), (i_3, j_5), (k, j_1), (k, j_4)$. But observe that X' still has an augmenting cycle at level 2 for node k: $k \to j_5 \to i_3 \to j_4 \to k$. However, with the original definition of $H_{i,\alpha}$, which links $j_2 \to t$, the maximum flow consists of pushing flow along the paths $k \to j_1 \to i_1 \to j_2 \to t$ and $k \to j_5 \to i_3 \to j_4 \to i_2 \to j_3 \to t$, leading to the new assignment $X'' = (i_1, j_2), (i_2, j_3), (i_3, j_4), (k, j_1), (k, j_5)$ which has no remaining augmenting cycles for k.

The Pareto-optimality of the assignment X^* returned by the algorithm follows from the following two claims.

Proposition 4.1. *There is no augmenting path after graph H is executed in Step 3 of* Pareto-Assignment.

Proposition 4.2. *There is no augmenting cycle associated with i (resp. j) at level α (resp. β) after graph $H_{i,\alpha}$ (resp. $H_{j,\beta}$) is executed in step 3 of* Pareto-Assignment.

Together, these two propositions imply that the outcome returned by Pareto-Assignment is indeed a Pareto-optimal assignment. Note that the construction of each graph $H, H_{i,\alpha}$ and $H_{j,\beta}$ is in polynomial time. In total there are $O(n^2)$ such graphs with $O(n^2)$ vertices each. For each graph $H, H_{i,\alpha}$ and $H_{j,\beta}$, its maximum flow can be computed in strongly polynomial time $O(n^6)$ with respect to its number of vertices $O(n^2)$ [18]. Therefore, the running time of the algorithm is in $O(n^8)$. This gives us our main result:

Theorem 4.1. *Algorithm* Pareto-Assignment *computes a Pareto-stable assignment in strongly polynomial time $O(n^8)$, where n is the total number of nodes in the bipartite graph M.*

5 Remarks

In one-to-one matching, the solution concepts of pairwise stability, core, and setwise stability all coincide, but this is not the case with many-to-many matching [27]. For our preference model for many-to-many matching, the core is not a suitable solution concept, since matchings in the core need not be pairwise stable ([28], Fig.1a), and the strong core need not exist ([27], §5.7) (it is easy to adapt the examples in these references to our model of preferences over sets). We also note that Pareto stability is incomparable with set-wise stability (both of which are strictly stronger than pairwise stability) in the sense that neither solution concept is stronger than the other— an easy example shows set-wise stable matchings need not be Pareto-optimal, and vice versa. The problem of computing set-wise, rather than pairwise, stable matchings, appears to be a challenging algorithmic question, and we leave it as an open problem for future work.

Acknowledgments. We are very grateful to Fuhito Kojima for helpful discussions and pointers to relevant literature. We also thank Mohammad Mahdian, David Pennock and Michael Schwarz for useful comments.

References

[1] A. Abdulkadiroglu, P. Pathak, A. E. Roth, *Strategy-proofness versus Efficiency in Matching with Indifferences: Redesigning the NYC High School Match*, American Economic Review, V.99(5), 1954-1978, 2009.

[2] A. Abdulkadiroglu, T. Sonmez. *School Choice: A Mechanism Design Approach*, American Economic Review, V.93(3), 729-747, 2003.

[3] D. Abraham, *Algorithmics of Two-Sided Matching Problems*, M.S. Thesis, University of Glasgow, 2003.

[4] N. Chen, A. Ghosh, N. Lambert, *Social Lending*, EC 2009, 335-344.

[5] J. Eeckhout, *On the Uniqueness of Stable Marriage Matchings*, Economics Letters, V.69, 1-8, 2000.

[6] F. Echenique, J. Oviedo, *A Theory of Stability in Many-to-many Matching Markets*, Theoretical Economics. Volume 1, Issue 2, June 2006, Pages 233-273.

[7] A. Erdil, *Two-sided Matching with Ties*, Ph.D. Dissertation, University of Chicago, 2006.

[8] A. Erdil, H. Ergin, *What's the Matter with Tie-breaking? Improving Efficiency in School Choice*, American Economic Review, V.98(3), 669-689, 2008.

[9] A. Erdil, H. Ergin, *Two-Sided Matching with Indifferences*, working paper.

[10] D. Gale, L. S. Shapley, *College Admissions and the Stability of Marriage*, American Mathematical Monthly, V.69, 9-15, 1962.

[11] D. Gusfield, R. W. Irving. *The Stable Marriage Problem: Structure and Algorithms*, MIT Press, 1989.

[12] J. Hatfield, S. Kominers, *Matching in Networks with Bilateral Contracts*, Proceedings of the 11th ACM Conference on Electronic Commerce, 2010.

[13] S. Herrero-Lopez, *Social Interactions in P2P Lending*, SNA-KDD Workshop on Social Network Mining and Analysis, 2009.

[14] K. Iwama, S. Miyazaki, *Stable Marriage with Ties and Incomplete Lists*, Encyclopedia of Algorithms, 2008.

[15] R. W. Irving, *Stable Marriage and Indifference*, Discrete Applied Mathematics, V.48, 261-272, 1994.

[16] K. Iwama, D. Manlove, S. Miyazaki, Y. Morita, *Stable Marriage with Incomplete Lists and Ties*, ICALP 1999, 443-452.

[17] T. Kavitha, K. Mehlhorn, D. Michail, K. E. Paluch, *Strongly Stable Matchings in Time O(nm) and Extension to the Hospitals-Residents Problem*, ACM Transactions on Algorithms, V.3(2), 2007.

[18] J. Kleinberg, E. Tardos, *Algorithm Design*, Addison Wesley, 2005.

[19] V. S. Malhotra, *On the Stability of Multiple Partner Stable Marriages with Ties*, ESA 2004, 508-519.

[20] D. F. Manlove, R. W. Irving, K. Iwama, S. Miyazaki, Y. Morita, *Hard Variants of Stable Marriage*, Theoretical Computer Science, V.276(1-2), 261-279, 2002.

[21] M. Niederle, L. Yariv, *Decentralized Matching with Aligned Preferences*, working paper.

[22] A. Ockenfels, A. E. Roth, *Late and Multiple Bidding in Second-Price Internet Auctions: Theory and Evidence Concerning Different Rules for Ending an Auction*, Games and Economic Behavior, V.55, 297-320, 2006.

[23] A. E. Roth, *The Evolution of the Labor Market for Medical Interns and Residents: a Case Study in Game Theory*, Journal of Political Economy, V.92(6), 991-1016, 1984.

[24] A. E. Roth, *The Economist as Engineer: Game Theory, Experimentation, and Computation as Tools for Design Economics*, Econometrica, V.70(4), 1341-1378, 2002.

[25] A. E. Roth, *Deferred Acceptance Algorithms: History, Theory, Practice, and Open Questions*, International Journal of Game Theory, 537-569, 2008.

[26] A. E. Roth, A. Ockenfels, *Last-Minute Bidding and the Rules for Ending Second-Price Auctions: Evidence from eBay and Amazon Auctions on the Internet*, American Economic Review, V.92(4), 1093-1103, 2002.

[27] A. E. Roth, M. Sotomayor, *Two-Sided Matching: A Study in Game-Theoretic Modeling and Analysis*, Cambridge University Press, 1992.

[28] M. Sotomayor, *Three Remarks on the Many-to-Many Stable Matching Problem*, Mathematical Social Sciences, V.38(1), 55-70, 1999.

[29] M. Sotomayor, *The Pareto-Stability Concept is a Natural Solution Concept*, working paper.

[30] H. Yanagisawa, *Approximation Algorithms for Stable Marriage Problems*, Ph.D. Thesis, Kyoto University, 2007.

Ning Chen
Division of Mathematical Sciences,
Nanyang Technological University,
Singapore.
Email: `ningc@ntu.edu.sg`

Arpita Ghosh
Yahoo! Research,
Santa Clara, CA, USA.
Email: `arpita@yahoo-inc.com`