Journal of the Operations Research Society of Japan
Vol. 59, No. 3, July 2016, pp. 225–240

MANY-TO-MANY STABLE MATCHINGS WITH TIES IN TREES

Keita Nakamura* Naoyuki Kamiyama
Kyushu University

(Received February 16, 2015; Revised January 22, 2016)

Abstract In the stable matching problem introduced by Gale and Shapley, it is known that in the case where the preference lists may involve ties, a stable matching always exists, but the sizes of stable matchings may be different. In this paper, we consider the problem of finding a maximum-size stable matching in a many-to-many matching market with ties. It is known that this problem is **NP**-hard even if the capacity of every agent is one. In this paper, we prove that this problem in trees can be solved in polynomial time by extending the algorithm proposed by Tayu and Ueno for the one-to-one setting.

Keywords: Discrete optimization, stable matching, tree, tie

1. Introduction

In the stable matching problem introduced by Gale and Shapley [1], there are two groups of agents such that each agent has a preference list over the members in the other group. The goal is to find a stable one-to-one matching. A matching is said to be stable, if there is no pair of agents that have incentive to break away from the current matching. If the preference lists do not involve a tie, then a stable matching always exists [1] and all stable matchings are of the same size [2]. In the case where the preference lists may involve ties, a stable matching[†] always exists [3] (we can prove this by breaking ties arbitrarily and using the result of [1]), but the sizes of stable matchings may be different [8]. In practical settings, the preference lists may indeed involve ties, and it is desirable to find a maximum-size stable matching [8]. Unfortunately, it is known [8] that the problem of finding a maximum-size stable matching is **NP**-hard. For such a hard problem, approximation algorithms and finding tractable cases may be escapes from its intractability. Several approximation algorithms for this problem were proposed, and the current best approximation ratio is 1.5 due to [5, 9, 10]. In this paper, we focus on the other approach, that is, finding tractable cases. For example, Irving, Manlove, and O'Malley [4] considered the case in which the lengths of the preference lists are bounded. Furthermore, Tayu and Ueno [11] proved that if an underlying bipartite graph is a tree, then this problem can be solved in linear time.

In this paper, we consider the many-to-many generalization of the above problem. That is, our goal is to find a maximum-size stable matching in a many-to-many matching market with ties. In our problem, each agent v has a capacity $q(v)$ and is allowed to be matched with at most $q(v)$ partners. We will consider this problem in trees, and give a polynomial-time algorithm for this case by extending the algorithm of Tayu and Ueno [11] for the one-to-one setting. To the best of our knowledge, our result is the first polynomial-time solvable case

*This work was done when Keita Nakamura was a student at Kyushu University.
[†]We adopt the stability criterion called the weak stability. See [7, Chapter 3] for a survey of stable matchings with ties.

of the maximum-size stable matching problem in a many-to-many matching market with ties (to the best of our knowledge, it is still open that the result of [4] can be generalized to the many-to-one setting).

The rest of this paper is organized as follows. In Section 2, we give a formal definition of our problem. In Section 3, we give characterizations of many-to-many stable matchings with ties in trees. In Section 4, we propose our algorithm based on the characterizations in Section 3. In Section 5, we give a faster implementation of our algorithm.

2. Preliminaries

The Many-to-Many Stable Matching with Ties problem (MMSMT for short) in trees is defined as follows. In this problem, we are given a tree $T = (V, E)$ that consists of a vertex set V and an edge set E. If there is an edge in E between distinct vertices u, v in V, then we denote by (u, v) this edge. Notice that (u, v) and (v, u) represent the same edge. For each vertex v in V, we denote by $N(v)$ the set of vertices w in V such that $(v, w) \in E$. For each vertex v in V and each subset F of E, let $\delta_F(v)$ be the set of edges in F incident to v.

For each vertex v in V, we are given a reflexive, transitive, and complete[‡] binary relation $\succsim_v$ on $N(v)$. For each vertex v in V, the binary relation $\succsim_v$ represents the preference list of v over $N(v)$. If $u \succsim_v w$ for a vertex v in V and vertices u, w in $N(v)$, then v prefers u to w, or is indifferent between u and w. Notice that there is a possibility that $u \succsim_v w$ and $w \succsim_v u$ for a vertex v in V and distinct vertices u, w in $N(v)$. For each vertex v in V and each pair of vertices u, w in $N(v)$, we write $u \succ_v w$ (resp., $u \sim_v w$), if $u \succsim_v w$ and $w \not\succsim_v u$ (resp., $u \succsim_v w$ and $w \succsim_v u$).

For each vertex v in V, we are given a positive integer $q(v)$ that represents the capacity of v. A subset M of E is called a *matching* in T, if $|\delta_M(v)| \leq q(v)$ for every vertex v in V. For each subset F of E and each edge (v, w) in $E \setminus F$, we say that (v, w) is *dominated* by F on v, if $|\delta_F(v)| = q(v)$ and $w' \succsim_v w$ for every edge (v, w') in $\delta_F(v)$. For each subset F of E and each edge e in $E \setminus F$, if e is dominated by F on at least one end vertex of e, then we simply say that e is dominated by F. A matching M in T is called a *stable matching* in T, if every edge in $E \setminus M$ is dominated by M. It is well known (see, e.g., [7, Section 5.4.3]) that if there are no distinct vertices u, v, w in V such that u, w in $N(v)$ and $u \sim_v w$, then there always exists a stable matching in T. Thus, in the same way of [3] for the one-to-one setting (i.e., by breaking ties arbitrarily), we can prove that there always exists a stable matching in T. We denote by $\mathbf{M}$ the set of stable matchings in T. Then, the goal of MMSMT in trees is to find a maximum-size stable matching in T.

2.1. Notation

We specify an arbitrary vertex r in V as the root of T, and regard T as a rooted tree with the root r. Without loss of generality, we can assume that $|\delta_E(r)| = 1$. We denote by (r, c_r) the unique edge in $\delta_E(r)$, and define $U := V \setminus \{r\}$. For each vertex v in U, we define

- $p_v :=$ the parent of v,
- $C_v :=$ the set of the children of v,
- $D_v :=$ the edge set of the subgraph of T induced by the descendants of v (including v),
- $S_v := D_v \cup \{(v, p_v)\}$.

For each vertex v in U and each matching M in T, we say that M is v-*stable*, if every edge in $D_v \setminus M$ is dominated by M. For each vertex v in U, we denote by $\mathbf{M}(v)$ the set of v-stable matchings M in T such that $M \subseteq S_v$. For each vertex v in U, we define

[‡]For every pair of vertices u, w in $N(v)$, at least one of $u \succsim_v w$ and $w \succsim_v u$ holds.

- $\mathbf{M}_P(v) := \{M \in \mathbf{M}(v) \mid (v, p_v) \in M\}$,
- $\mathbf{M}_{\overline{P}}(v) := \{M \in \mathbf{M}(v) \mid (v, p_v) \notin M\}$,
- $\mathbf{M}_F(v) := \{M \in \mathbf{M}_{\overline{P}}(v) \mid |\delta_M(v)| = q(v)$ and $c \succsim_v p_v$ for every edge (v, c) in $\delta_M(v)\}$.

It is not difficult to see that for every vertex v in U and every subset M of D_v, $M \in \mathbf{M}_{\overline{P}}(v)$ if and only if M is a stable matching in the subgraph of T induced by the descendants of v. Thus, for every vertex v in U, since there always exists a stable matching in the subgraph of T induced by the descendants of v, $\mathbf{M}_{\overline{P}}(v)$ is not empty. For each vertex v in U, we define

- $\mathbf{M}_P^{=}(v) := \{M \in \mathbf{M}_P(v) \mid |\delta_M(v)| = q(v)\}$,
- $\mathbf{M}_P^{<}(v) := \{M \in \mathbf{M}_P(v) \mid |\delta_M(v)| < q(v)\}$,
- $\mathbf{M}_{\overline{P}}^{=}(v) := \{M \in \mathbf{M}_{\overline{P}}(v) \mid |\delta_M(v)| = q(v)\}$,
- $\mathbf{M}_{\overline{P}}^{<}(v) := \{M \in \mathbf{M}_{\overline{P}}(v) \mid |\delta_M(v)| < q(v)\}$.

For each vertex v in U, we partition C_v into $C_v^1, C_v^2, \ldots, C_v^{\ell(v)}$ in such a way that

- $u \sim_v w$ for every $i = 1, 2, \ldots, \ell(v)$ and every pair of vertices u, w in C_v^i, and
- $u \succ_v w$ for every pair of $i, j = 1, 2, \ldots, \ell(v)$ such that $i < j$ and every pair of vertices u in C_v^i and w in C_v^j.

For each vertex v in U and each $i = 1, 2, \ldots, \ell(v)$, we define

$$K_v^i := C_v^1 \cup C_v^2 \cup \cdots \cup C_v^i.$$

In addition, for each vertex v in U, define $K_v^0 := \emptyset$, $C_v^{\ell(v)+1} := \emptyset$, and $K_v^{\ell(v)+1} := C_v$. For each vertex v in U, let $t^+(v)$ be the minimum integer i in $\{1, 2, \ldots, \ell(v)\}$ such that $p_v \succsim_v c$ for a vertex c in C_v^i. If $c \succ_v p_v$ for every vertex c in C_v, then we define $t^+(v) := \ell(v) + 1$. For each vertex v in U, we denote by $t^-(v)$ the maximum integer i in $\{1, 2, \ldots, \ell(v)\}$ such that $c \succsim_v p_v$ for a vertex c in C_v^i. If $p_v \succ_v c$ for every vertex c in C_v, then we define $t^-(v) := 0$.

3. Characterizations

In this section, we prove lemmas that will be needed in the next section.

We first prove the following two easy lemmas.

Lemma 3.1. *For every subset M of E, $M \in \mathbf{M}$ if and only if $M \in \mathbf{M}_P(c_r) \cup \mathbf{M}_F(c_r)$.*

Proof. Let M be a subset of E. We first prove the *if* part. Since M is a c_r-stable matching in T, M is a matching in T. Furthermore, the c_r-stability of M implies that M dominates every edge in $E \setminus (M \cup \{(c_r, r)\})$. If $M \in \mathbf{M}_P(c_r)$, then $(c_r, r) \in M$. In addition, if $M \in \mathbf{M}_F(c_r)$, then (c_r, r) is dominated by M on c_r. This completes the proof of the *if* part.

Next we prove the *only if* part. Since M is a stable matching in T, M is also a c_r-stable matching in T. Furthermore, since $S_{c_r} = E$, $M \in \mathbf{M}(c_r)$. If $(c_r, r) \in M$, then $M \in \mathbf{M}_P(c_r)$. Assume that $(c_r, r) \notin M$. If $M \in \mathbf{M}_{\overline{P}}(c_r) \setminus \mathbf{M}_F(c_r)$, then (c_r, r) is dominated by M on neither c_r nor r, which contradicts the fact that $M \in \mathbf{M}$. Thus, $M \in \mathbf{M}_F(c_r)$. □

Lemma 3.2. *Let v be a vertex in U, and let M be a subset of S_v. If M is a v-stable matching in T, then for every vertex c in C_v, $M \cap S_c$ is a c-stable matching in T.*

Proof. Assume that M is a v-stable matching in T. Then, for every vertex c in C_v, every edge in $D_c \setminus M$ is dominated by $M \cap S_c$ and $M \cap S_c$ is a matching in T. This completes the proof. □

In what follows, we give characterizations of members in $\mathbf{M}_P^=(v)$, $\mathbf{M}_P^<(v)$, $\mathbf{M}_{\overline{P}}^=(v)$, $\mathbf{M}_{\overline{P}}^<(v)$, and $\mathbf{M}_F(v)$ for each vertex v in U. Although these characterizations are natural generalizations of the characterizations in [11] to the many-to-many setting, there is the following difference. In several characterizations of [11] for a vertex v in U, we choose a "key" child of v to which v should be connected, and then we categorize the children of v based on $\succsim_v$ and this key child. On the other hand, in our characterizations, we categorize the children of v without choosing such a key child.

Lemma 3.3. *For every vertex v in U and every subset M of S_v, $M \in \mathbf{M}_P^=(v)$ if and only if*

1. $|\delta_M(v)| = q(v)$,
2. $(v, p_v) \in M$, *and*
3. *there is an integer ξ in $\{t^+(v), t^+(v)+1, \ldots, \ell(v)+1\}$ such that*
   - $M \cap S_c \in \mathbf{M}_P(c) \cup \mathbf{M}_F(c)$ *for every vertex c in $K_v^{\xi-1}$,*
   - $M \cap S_c \in \mathbf{M}_P(c) \cup \mathbf{M}_{\overline{P}}(c)$ *for every vertex c in C_v^{ξ}, and*
   - $M \cap S_c \in \mathbf{M}_{\overline{P}}(c)$ *for every vertex c in $C_v \setminus K_v^{\xi}$.*

Proof. Assume that we are given a vertex v in U and a subset M of S_v. We first prove the *if* part. Since $q(p_v) > 0$, the conditions 1, 2, and 3 imply that M is a matching in T such that $|\delta_M(v)| = q(v)$ and $(v, p_v) \in M$. Thus, what remains is to prove that M is v-stable. The condition 3 implies that for every vertex c in C_v, $M \cap S_c$ is a c-stable matching in T, i.e., every edge in $D_c \setminus M$ is dominated by M. Thus, it suffices to prove that for every vertex c in C_v, $(v, c) \in M$ or (v, c) is dominated by M.

We first consider a vertex c in $K_v^{\xi-1}$. If $M \cap S_c \in \mathbf{M}_P(c)$, then $(v, c) \in M$. In addition, if $M \cap S_c \in \mathbf{M}_F(c)$, then (v, c) is dominated by M on c. Next we consider a vertex c in C_v^{ξ}. The condition 3 and $\xi \geq t^+(v)$ imply that $c' \succsim_v c$ for every edge (v, c') in $\delta_M(v)$. Thus, since the condition 1 implies that $|\delta_M(v)| = q(v)$, $(v, c) \in M$ or (v, c) is dominated by M on v. Lastly, we consider a vertex c in $C_v \setminus K_v^{\xi}$. The condition 3 and $\xi \geq t^+(v)$ imply that $c' \succ_v c$ for every edge (v, c') in $\delta_M(v)$. Thus, since the condition 1 implies that $|\delta_M(v)| = q(v)$, (v, c) is dominated by M on v. This completes the proof of the *if* part.

Next we prove the *only if* part. Since the definition of $\mathbf{M}_P^=(v)$ implies that the conditions 1 and 2 hold, it suffices to prove that the condition 3 holds. Let ξ' be the maximum integer i in $\{1, 2, \ldots, \ell(v)\}$ such that $(v, c) \in M$ for some vertex c in C_v^i. If there is no vertex c in C_v such that $(v, c) \in M$, then we define $\xi' := 0$. We define $\xi := \max\{t^+(v), \xi'\}$, and prove that ξ satisfies the condition 3. Lemma 3.2 implies that for every vertex c in C_v, $M \cap S_c$ is a c-stable matching in T. Thus, what remains is to prove that for every vertex c in C_v, $M \cap S_c$ is in an appropriate family of subsets of S_c. Define c^* as follows. If $\xi' \leq t^+(v)$, then we define $c^* := p_v$. If $\xi' > t^+(v)$, then we define c^* as a vertex c in $C_v^{\xi'}$ such that $(v, c) \in M$.

We first consider a vertex c in $K_v^{\xi-1}$. If $M \cap S_c$ is in $\mathbf{M}_{\overline{P}}(c) \setminus \mathbf{M}_F(c)$, then (v, c) is not dominated by M on c. In addition, since $c \succ_v c^*$ and $(v, c^*) \in M$, (v, c) is not dominated by M on v, which contradicts the fact that M is a v-stable matching in T. These observations imply that $M \cap S_c \in \mathbf{M}_P(c) \cup \mathbf{M}_F(c)$. For every vertex c in C_v^{ξ}, since $M \cap S_c$ is a c-stable matching in T, the proof is done. Let c be a vertex in $C_v \setminus K_v^{\xi}$. The definition of ξ implies that $(v, c) \notin M$. Thus, since $M \cap S_c$ is a c-stable matching in T, $M \cap S_c \in \mathbf{M}_{\overline{P}}(c)$. $\square$

Lemma 3.4. *For every vertex v in U and every subset M of S_v, $M \in \mathbf{M}_P^<(v)$ if and only if*

1. $|\delta_M(v)| < q(v)$,
2. $(v, p_v) \in M$, *and*
3. $M \cap S_c \in \mathbf{M}_P(c) \cup \mathbf{M}_F(c)$ *for every vertex c in C_v.*

Proof. Assume that we are given a vertex v in U and a subset M of S_v. We first prove the *if* part. The conditions 1, 2, and 3 imply that M is a matching in T such that $|\delta_M(v)| < q(v)$ and $(v, p_v) \in M$. Thus, what remains is to prove that M is v-stable. Since $M \cap S_c$ is a c-stable matching in T for every vertex c in C_v, it suffices to prove that $(v, c) \in M$ or (v, c) is dominated by M for every vertex c in C_v. For every vertex c in C_v such that $(v, c) \notin M$, since $M \in \mathbf{M}_F(c)$, (v, c) is dominated by M on c. This completes the proof of the *if* part.

Next we prove the *only if* part. Since the definition of $\mathbf{M}_P^{<}(v)$ implies that the conditions 1 and 2 hold, it suffices to prove that the condition 3 holds. Lemma 3.2 implies that $M \cap S_c$ is a c-stable matching in T for every vertex c in C_v. If $M \cap S_c \in \mathbf{M}_{\overline{P}}(c) \setminus \mathbf{M}_F(c)$, then since $|\delta_M(v)| < q(v)$ follows from the definition of $\mathbf{M}_P^{<}(v)$, (v, c) is dominated by M on neither v nor c. Thus, $M \cap S_c \in \mathbf{M}_P(c) \cup \mathbf{M}_F(c)$. □

Lemma 3.5. *For every vertex v in U and every subset M of S_v, $M \in \mathbf{M}_{\overline{P}}^{=}(v)$ if and only if*

1. *$|\delta_M(v)| = q(v)$,*
2. *$(v, p_v) \notin M$, and*
3. *there is an integer ξ in $\{1, 2, \ldots, \ell(v)\}$ such that*
   - *$M \cap S_c \in \mathbf{M}_P(c) \cup \mathbf{M}_F(c)$ for every vertex c in $K_v^{\xi-1}$,*
   - *$M \cap S_c \in \mathbf{M}_P(c) \cup \mathbf{M}_{\overline{P}}(c)$ for every vertex c in C_v^{ξ}, and*
   - *$M \cap S_c \in \mathbf{M}_{\overline{P}}(c)$ for every vertex c in $C_v \setminus K_v^{\xi}$.*

Proof. Assume that we are given a vertex v in U and a subset M of S_v. We first prove the *if* part. The conditions 1, 2, and 3 imply that M is a matching in T such that $|\delta_M(v)| = q(v)$ and $(v, p_v) \notin M$. What remains is to prove that M is v-stable. The condition 3 implies that for every vertex c in C_v, every edge in $D_c \setminus M$ is dominated by M. Thus, it suffices to prove that for every vertex c in C_v, $(v, c) \in M$ or (v, c) is dominated by M.

We first consider a vertex c in $K_v^{\xi-1}$. If $M \cap S_c \in \mathbf{M}_P(c)$, then $(v, c) \in M$. In addition, if $M \cap S_c \in \mathbf{M}_F(c)$, then (v, c) is dominated by M on c. Next we consider a vertex c in C_v^{ξ}. The condition 3 implies that $c' \succsim_v c$ for every edge (v, c') in $\delta_M(v)$. Thus, since the condition 1 implies that $|\delta_M(v)| = q(v)$, $(v, c) \in M$ or (v, c) is dominated by M on v. Lastly we consider a vertex c in $C_v \setminus K_v^{\xi}$. The condition 3 implies that $c' \succ_v c$ for every edge (v, c') in $\delta_M(v)$. Thus, since the condition 1 implies that $|\delta_M(v)| = q(v)$, (v, c) is dominated by M on v. This completes the proof of the *if* part.

Next we prove the *only if* part. The definition of $\mathbf{M}_{\overline{P}}^{=}(v)$ implies the conditions 1 and 2, and thus it suffices to prove that the condition 3 holds. Since $|q(v)| > 0$, there is an integer i in $\{1, 2, \ldots, \ell(v)\}$ such that $(v, c) \in M$ for some vertex c in C_v^i. Let ξ be the maximum integer i in $\{1, 2, \ldots, \ell(v)\}$ such that $(v, c) \in M$ for some vertex c in C_v^i. We will prove that ξ satisfies the condition 3. Lemma 3.2 implies that for every vertex c in C_v, $M \cap S_c$ is a c-stable matching in T. What remains is to prove that $M \cap S_c$ is in an appropriate family of subsets of S_c for every vertex c in C_v. Let c^* be a vertex c in C_v^{ξ} such that $(v, c) \in M$.

We first consider a vertex c in $K_v^{\xi-1}$. If $M \cap S_c$ is in $\mathbf{M}_{\overline{P}}(c) \setminus \mathbf{M}_F(c)$, then (v, c) is not dominated by M on c. Furthermore, since $c \succ_v c^*$ and $(v, c^*) \in M$, (v, c) is not dominated by M on v, which contradicts the fact that M is a v-stable matching in T. Thus, $M \cap S_c$ is in $\mathbf{M}_P(c) \cup \mathbf{M}_F(c)$. For every vertex c in C_v^{ξ}, since $M \cap S_c$ is a c-stable matching in T, the proof is done. Let c be a vertex in $C_v \setminus K_v^{\xi}$. The definition of c^* implies that $(v, c) \notin M$. Thus, since $M \cap S_c$ is a c-stable matching in T, $M \cap S_c \in \mathbf{M}_{\overline{P}}(c)$. □

Lemma 3.6. *For every vertex v in U and every subset M of S_v, $M \in \mathbf{M}_{\overline{P}}^{<}(v)$ if and only if*

1. *$|\delta_M(v)| < q(v)$,*
2. *$(v, p_v) \notin M$, and*

3. $M \cap S_c \in \mathbf{M}_P(c) \cup \mathbf{M}_F(c)$ *for every vertex* c *in* C_v.

Proof. Assume that we are given a vertex v in U and a subset M of S_v. We first prove the *if* part. The conditions 1, 2, and 3 imply that M is a matching in T such that $|\delta_M(v)| < q(v)$ and $(v, p_v) \notin M$. What remains is to prove that M is v-stable. Since $M \cap S_c$ is a c-stable matching in T for every vertex c in C_v, it is sufficient to prove that $(v, c) \in M$ or (v, c) is dominated by M for every vertex c in C_v. For every vertex c in C_v such that $(v, c) \notin M$, since $M \in \mathbf{M}_F(c)$, (v, c) is dominated by M on c. This completes the proof of the *if* part.

Next we prove the *only if* part. Since the definition of $\mathbf{M}^{\leq}_{\overline{P}}(v)$ implies that the conditions 1 and 2 hold, it suffices to prove that the condition 3 holds. Lemma 3.2 implies that $M \cap S_c$ is a c-stable matching in T for every vertex c in C_v. If $M \cap S_c \in \mathbf{M}_{\overline{P}}(c) \setminus \mathbf{M}_F(c)$, then since $|\delta_M(v)| < q(v)$ follows from the definition of $\mathbf{M}^{\leq}_{\overline{P}}(v)$, (v, c) is dominated by M on neither v nor c. Thus, $M \cap S_c \in \mathbf{M}_P(c) \cup \mathbf{M}_F(c)$. □

Lemma 3.7. *For every vertex* v *in* U *and every subset* M *of* S_v, $M \in \mathbf{M}_F(v)$ *if and only if*

1. $|\delta_M(v)| = q(v)$,
2. $(v, p_v) \notin M$, *and*
3. $t^-(v) > 0$ *and there is an integer* ξ *in* $\{1, 2, \ldots, t^-(v)\}$ *such that*
   - $M \cap S_c \in \mathbf{M}_P(c) \cup \mathbf{M}_F(c)$ *for every vertex* c *in* $K_v^{\xi-1}$,
   - $M \cap S_c \in \mathbf{M}_P(c) \cup \mathbf{M}_{\overline{P}}(c)$ *for every vertex* c *in* C_v^{ξ}, *and*
   - $M \cap S_c \in \mathbf{M}_{\overline{P}}(c)$ *for every vertex* c *in* $C_v \setminus K_v^{\xi}$.

Proof. Assume that we are given a vertex v in U and a subset M of S_v. We first prove the *if* part. The conditions 1, 2, and 3 imply that M is a matching in T such that $|\delta_M(v)| = q(v)$ and $(v, p_v) \notin M$. Furthermore, since $\xi \in \{1, 2, \ldots, t^-(v)\}$ and $M \cap S_c \in \mathbf{M}_{\overline{P}}(c)$ for every vertex c in $C_v \setminus K_v^{\xi}$, we have $c \succsim_v p_v$ for every edge (v, c) in $\delta_M(v)$. Thus, what remains is to prove that M is v-stable. The condition 3 implies that for every vertex c in C_v, every edge in $D_c \setminus M$ is dominated by M. Thus, it suffices to prove that for every vertex c in C_v, $(v, c) \in M$ or (v, c) is dominated by M.

We first consider a vertex c in $K_v^{\xi-1}$. If $M \cap S_c \in \mathbf{M}_P(c)$, then $(v, c) \in M$. Furthermore, if $M \cap S_c \in \mathbf{M}_F(c)$, then (v, c) is dominated by M on c. Next we consider a vertex c in C_v^{ξ}. The condition 3 implies that $c' \succsim_v c$ for every edge (v, c') in $\delta_M(v)$. Thus, since the condition 1 implies that $|\delta_M(v)| = q(v)$, $(v, c) \in M$ or (v, c) is dominated by M on v. Lastly we consider a vertex c in $C_v \setminus K_v^{\xi}$. The condition 3 implies that $c' \succ_v c$ for every edge (v, c') in $\delta_M(v)$. Thus, since the condition 1 implies that $|\delta_M(v)| = q(v)$, (v, c) is dominated by M on v. This completes the proof of the *if* part.

Next we prove the *only if* part. Since the definition of $\mathbf{M}_F(v)$ implies the conditions 1 and 2, it suffices to prove that the condition 3 holds. Since $|q(v)| > 0$, there is an integer i in $\{1, 2, \ldots, \ell(v)\}$ such that $(v, c) \in M$ for some vertex c in C_v^i. Let ξ be the maximum integer i in $\{1, 2, \ldots, \ell(v)\}$ such that $(v, c) \in M$ for some vertex c in C_v^i. Since $c \succsim_v p_v$ for every edge (v, c) in $\delta_M(v)$, $t^-(v) > 0$ and $\xi \in \{1, 2, \ldots, t^-(v)\}$. We will prove that ξ satisfies the condition 3. Lemma 3.2 implies that for every vertex c in C_v, $M \cap S_c$ is a c-stable matching in T. Thus, what remains is to prove that for every vertex c in C_v, $M \cap S_c$ is in an appropriate family of subsets of S_c. Let c^* be a vertex c in C_v^{ξ} such that $(v, c) \in M$.

We first consider a vertex c in $K_v^{\xi-1}$. If $M \cap S_c$ is in $\mathbf{M}_{\overline{P}}(c) \setminus \mathbf{M}_F(c)$, then (v, c) is not dominated by M on c. Furthermore, since $c \succ_v c^*$ and $(v, c^*) \in M$, (v, c) is not dominated by M on v, which contradicts the fact that M is a v-stable matching in T. Thus, $M \cap S_c$ is in $\mathbf{M}_P(c) \cup \mathbf{M}_F(c)$. For every vertex c in C_v^{ξ}, since $M \cap S_c$ is a c-stable matching in T, the proof is done. Let c be a vertex in $C_v \setminus K_v^{\xi}$. The definition of c^* implies that $(v, c) \notin M$. Thus, since $M \cap S_c$ is a c-stable matching in T, $M \cap S_c \in \mathbf{M}_{\overline{P}}(c)$. □

4. Algorithm

In this section, we give a polynomial-time algorithm for MMSMT in trees. We first concentrate on computing the maximum-size of a stable matching. We can easily modify our algorithm in such a way that it can find a maximum-size stable matching (see Section 5).

For each vertex v in U and each symbol X in $\{P, \overline{P}, F\}$, we define $\mu_X(v)$ by

$$\mu_X(v) := \begin{cases} \max\{|M| \mid M \in \mathbf{M}_X(v)\} & \text{if } \mathbf{M}_X(v) \neq \emptyset \\ -\infty & \text{if } \mathbf{M}_X(v) = \emptyset. \end{cases}$$

It is not difficult to see that for every leaf vertex v in U, we have

$$\mu_P(v) = 1, \quad \mu_{\overline{P}}(v) = 0, \quad \mu_F(v) = -\infty. \tag{4.1}$$

For each vertex v in U, we define the *depth* $d(v)$ of v as the number of edges of the unique path from r to v in T. Our algorithm can be described as follows.

Algorithm 1 Algorithm for MMSMT in trees.

1: Set $i := \max\{d(v) \mid v \in U\}$.
2: **while** $i \geq 1$ **do**
3:   **for** all vertices v in U such that $d(v) = i$ **do**
4:     **if** v is a leaf vertex **then**
5:       Compute $\mu_P(v)$, $\mu_{\overline{P}}(v)$, and $\mu_F(v)$ as (4.1).
6:     **else**
7:       Compute $\mu_P(v)$, $\mu_{\overline{P}}(v)$, and $\mu_F(v)$ by using $\mu_X(c)$ for vertices c in C_v and symbols X in $\{P, \overline{P}, F\}$.
8:     **end if**
9:   **end for**
10:   Set $i := i - 1$.
11: **end while**
12: Output $\max\{\mu_P(c_r), \mu_F(c_r)\}$, and halt.

For proving the correctness of Algorithm 1, Lemma 3.1 implies that it suffices to prove that in the line 7 of Algorithm 1, we can compute $\mu_P(v)$, $\mu_{\overline{P}}(v)$, and $\mu_F(v)$ by using $\mu_X(c)$ for vertices c in C_v and symbols X in $\{P, \overline{P}, F\}$. In what follows, we prove this by using the characterizations in Section 3.

Here we explain about relationship between the algorithm of [11] and our algorithm. The frameworks of these algorithms are the same. In the one-to-one setting (i.e., the algorithm of [11]), since the number of edges incident to each vertex is at most one, the implementation of the line 7 is simple. For extending their algorithm to the many-to-many setting, we have to modify this part to the many-to-many setting. This is our main contribution.

In what follows, we assume that we are given a non-leaf vertex v in U, and we know $\mu_X(c)$ for all vertices c in C_v and all symbols X in $\{P, \overline{P}, F\}$. Under this assumption, we consider how to compute $\mu_X(v)$ for all symbols X in $\{P, \overline{P}, F\}$.

4.1. Notation

Here we introduce notation that will be needed later. For each vertex c in C_v such that $\mu_P(c) \neq -\infty$ and $\mu_F(c) \neq -\infty$, we define

$$\varphi_F(c) := \mu_P(c) - \mu_F(c).$$

Furthermore, for each vertex c in C_v such that $\mu_P(c) \neq -\infty$, we define

$$\varphi_{\overline{P}}(c) := \mu_P(c) - \mu_{\overline{P}}(c).$$

Recall that $\mu_{\overline{P}}(c) \neq -\infty$ for every vertex c in C_v. For each member M in $\mathbf{M}(v)$, we define

$$\partial_v M := \{c \in C_v \mid (v, c) \in M\}.$$

For each $i = 1, 2, \ldots, \ell(v) + 1$, we define X_c^i by

$$X_c^i := \begin{cases} F & \text{if } c \in K_v^{i-1} \\ \overline{P} & \text{if } c \in C_v \setminus K_v^{i-1}. \end{cases}$$

For each $i = 1, 2, \ldots, \ell(v) + 1$, we define Z_P^i and Z_F^i by

$$Z_P^i := \{c \in K_v^i \mid \mu_P(c) \neq -\infty\}, \quad Z_F^i := \{c \in K_v^i \mid \mu_F(c) = -\infty\}.$$

Furthermore, we define $Z_F := \{c \in C_v \mid \mu_F(c) = -\infty\}$.

4.2. Algorithm for computing $\mu_P(v)$

Here we consider how to compute $\mu_P(v)$. We define $\mu_P^{=}(v)$ and $\mu_P^{<}(v)$ by

$$\mu_P^{=}(v) := \begin{cases} \max\{|M| \mid M \in \mathbf{M}_P^{=}(v)\} & \text{if } \mathbf{M}_P^{=}(v) \neq \emptyset \\ -\infty & \text{if } \mathbf{M}_P^{=}(v) = \emptyset, \end{cases}$$

$$\mu_P^{<}(v) := \begin{cases} \max\{|M| \mid M \in \mathbf{M}_P^{<}(v)\} & \text{if } \mathbf{M}_P^{<}(v) \neq \emptyset \\ -\infty & \text{if } \mathbf{M}_P^{<}(v) = \emptyset. \end{cases}$$

Clearly, we have $\mu_P(v) = \max\{\mu_P^{=}(v), \mu_P^{<}(v)\}$. Thus, it suffices to compute $\mu_P^{=}(v)$ and $\mu_P^{<}(v)$.

We first consider how to compute $\mu_P^{=}(v)$. For each $i = t^+(v), t^+(v) + 1, \ldots, \ell(v) + 1$, we denote by $\mathbf{M}_P^{=}(v, i)$ the set of members M in $\mathbf{M}_P^{=}(v)$ such that

- $M \cap S_c \in \mathbf{M}_P(c) \cup \mathbf{M}_F(c)$ for every vertex c in K_v^{i-1},
- $M \cap S_c \in \mathbf{M}_P(c) \cup \mathbf{M}_{\overline{P}}(c)$ for every vertex c in C_v^i, and
- $M \cap S_c \in \mathbf{M}_{\overline{P}}(c)$ for every vertex c in $C_v \setminus K_v^i$.

Furthermore, for each $i = t^+(v), t^+(v) + 1, \ldots, \ell(v) + 1$, we define $\mu_P^{=}(v, i)$ by

$$\mu_P^{=}(v, i) := \begin{cases} \max\{|M| \mid M \in \mathbf{M}_P^{=}(v, i)\} & \text{if } \mathbf{M}_P^{=}(v, i) \neq \emptyset \\ -\infty & \text{if } \mathbf{M}_P^{=}(v, i) = \emptyset. \end{cases}$$

Since Lemma 3.3 implies that $\mathbf{M}_P^{=}(v) = \bigcup_{i=t^+(v)}^{\ell(v)+1} \mathbf{M}_P^{=}(v, i)$, we have

$$\mu_P^{=}(v) = \max\{\mu_P^{=}(v, i) \mid i = t^+(v), t^+(v) + 1, \ldots, \ell(v) + 1\}.$$

Thus, it suffices to compute $\mu_P^{=}(v, i)$ for each $i = t^+(v), t^+(v) + 1, \ldots, \ell(v) + 1$.

Let i be an integer in $\{t^+(v), t^+(v) + 1, \ldots, \ell(v) + 1\}$. Then we consider how to compute $\mu_P^{=}(v, i)$. Lemma 3.3 implies the following lemma.

Lemma 4.1. *We have $\mu^{=}_{\overline{P}}(v,i) \neq -\infty$ if and only if*

- *there is no vertex c in K_v^{i-1} such that $\mu_P(c) = \mu_F(c) = -\infty$,*
- *there are at most $q(v)-1$ vertices c in K_v^{i-1} such that $\mu_F(c) = -\infty$, and*
- *there are at least $q(v)-1$ vertices c in K_v^i such that $\mu_P(c) \neq -\infty$.*

Lemma 4.1 implies that we can decide whether $\mu^{=}_{\overline{P}}(v,i) \neq -\infty$ by using $\mu_X(c)$ for vertices c in C_v and symbols X in $\{P, \overline{P}, F\}$. Thus, in what follows, we assume that $\mu^{=}_{\overline{P}}(v,i) \neq -\infty$. A subset Π of K_v^i is said to be $(\overline{P}, =, i)$-*feasible*, if

$$|\Pi| = q(v) - 1, \quad Z_F^{i-1} \subseteq \Pi \subseteq Z_P^i.$$

Let $\mathcal{F}^{=,i}_{\overline{P}}$ be the family of $(\overline{P}, =, i)$-feasible subsets of K_v^i. Then, Lemma 3.3 implies that

(F1) for every member M in $\mathbf{M}^{=}_{\overline{P}}(v,i)$, we have $\partial_v M \in \mathcal{F}^{=,i}_{\overline{P}}$, and

(F2) for every member Π in $\mathcal{F}^{=,i}_{\overline{P}}$, there is a member M in $\mathbf{M}^{=}_{\overline{P}}(v,i)$ such that $\partial_v M = \Pi$.

For each member Π in $\mathcal{F}^{=,i}_{\overline{P}}$, we define $\mathbf{M}^{=}_{\overline{P}}(v,i;\Pi)$ by

$$\mathbf{M}^{=}_{\overline{P}}(v,i;\Pi) := \{M \in \mathbf{M}^{=}_{\overline{P}}(v,i) \mid \partial_v M = \Pi\}.$$

Then, (F1) implies that $\mathbf{M}^{=}_{\overline{P}}(v,i) = \bigcup_{\Pi \in \mathcal{F}^{=,i}_{\overline{P}}} \mathbf{M}^{=}_{\overline{P}}(v,i;\Pi)$. In addition, (F2) implies that for every member Π in $\mathcal{F}^{=,i}_{\overline{P}}$, $\mathbf{M}^{=}_{\overline{P}}(v,i;\Pi)$ is not empty. Define $\mu^{=}_{\overline{P}}(v,i;\Pi)$ by

$$\mu^{=}_{\overline{P}}(v,i;\Pi) := \max\{|M| \mid M \in \mathbf{M}^{=}_{\overline{P}}(v,i;\Pi)\}.$$

Then, we have

$$\mu^{=}_{\overline{P}}(v,i) = \max\{\mu^{=}_{\overline{P}}(v,i;\Pi) \mid \Pi \in \mathcal{F}^{=,i}_{\overline{P}}\}.$$

Furthermore, it is not difficult to see that for each member Π in $\mathcal{F}^{=,i}_{\overline{P}}$,

$$\begin{aligned}
\mu^{=}_{\overline{P}}(v,i;\Pi) &= \sum_{c \in \Pi} \mu_P(c) + \sum_{c \in C_v \setminus \Pi} \mu_{X_c^i}(c) + 1 \\
&= \sum_{c \in \Pi \setminus Z_F^{i-1}} \mu_P(c) + \sum_{c \in Z_F^{i-1}} \mu_P(c) + \sum_{c \in C_v \setminus \Pi} \mu_{X_c^i}(c) + 1 \\
&= \sum_{c \in \Pi \setminus Z_F^{i-1}} \varphi_{X_c^i}(c) + \sum_{c \in Z_F^{i-1}} \mu_P(c) + \sum_{c \in C_v \setminus Z_F^{i-1}} \mu_{X_c^i}(c) + 1.
\end{aligned} \tag{4.2}$$

Thus, in order to compute $\mu^{=}_{\overline{P}}(v,i)$, it suffices to find Π that maximizes the first term in the last line of (4.2). This implies that we can compute $\mu^{=}_{\overline{P}}(v,i)$ by Algorithm 2.

Algorithm 2 Algorithm for computing $\mu^{=}_{\overline{P}}(v,i)$

Sort vertices in $Z_P^i \setminus Z_F^{i-1}$ as $c_1, c_2, \ldots$ is such a way that $\varphi_{X_{c_1}^i}(c_1) \geq \varphi_{X_{c_2}^i}(c_2) \geq \cdots$.
Set $\Pi := \{c_1, c_2, \ldots, c_{q(v)-1-|Z_F^{i-1}|}\} \cup Z_F^{i-1}$.
Output $\mu^{=}_{\overline{P}}(v,i;\Pi)$, and halt.

The following lemma follows from the above argument.

Lemma 4.2. *We can compute $\mu^{=}_{\overline{P}}(v,i)$ by using Algorithm 2.*

Next we consider how to compute $\mu^{<}_{\overline{P}}(v)$. Lemma 3.4 implies the following lemma.

Lemma 4.3. *We have $\mu^{<}_{\overline{P}}(v) \neq -\infty$ if and only if*

- *there is no vertex c in C_v such that $\mu_P(c) = \mu_F(c) = -\infty$, and*

- *there are at most $q(v)-2$ vertices in C_v such that $\mu_F(c) = -\infty$.*

Lemma 4.3 implies that we can decide whether $\mu_P^{<}(v) \neq -\infty$ by using $\mu_X(c)$ for vertices c in C_v and symbols X in $\{P, \overline{P}, F\}$. Thus, in what follows, we assume that $\mu_P^{<}(v) \neq -\infty$. A subset Π of C_v is said to be $(P,<)$-*feasible*, if

$$|\Pi| \leq q(v) - 2, \qquad Z_F \subseteq \Pi.$$

Then, Lemma 3.4 implies that
- for every member M in $\mathbf{M}_P^{<}(v)$, $\partial_v M$ is a $(P,<)$-feasible subset of C_v, and
- for every $(P,<)$-feasible subset Π of C_v, there exists a member M in $\mathbf{M}_P^{<}(v)$ such that $\partial_v M = \Pi$.

For each $(P,<)$-feasible subset Π of C_v, we define $\mu_P^{<}(v;\Pi)$ by

$$\mu_P^{<}(v;\Pi) := \max\{|M| \mid M \in \mathbf{M}_P^{<}(v),\ \partial_v M = \Pi\}.$$

Then, for each $(P,<)$-feasible subset Π of C_v,

$$\mu_P^{<}(v;\Pi) = \sum_{c \in \Pi \setminus Z_F} \varphi_F(c) + \sum_{c \in Z_F} \mu_P(c) + \sum_{c \in C_v \setminus Z_F} \mu_F(c) + 1. \tag{4.3}$$

In order to compute $\mu_P^{<}(v)$, it suffices to find Π that maximizes the first term of (4.3). Thus, we can compute $\mu_P^{<}(v)$ by Algorithm 3.

Algorithm 3 Algorithm for computing $\mu_P^{<}(v)$

Sort vertices in $C_v \setminus Z_F$ as $c_1, c_2, \ldots$ in such a way that $\varphi_F(c_1) \geq \varphi_F(c_2) \geq \cdots$.
if $\varphi_F(c_1) \leq 0$ **then**
    Set $\Pi := \emptyset$.
else
    Set ξ to be the maximum integer j in $\{1, 2, \ldots, |C_v \setminus Z_F|\}$ such that $\varphi_F(c_j) > 0$.
    Set $\Pi := \{c_1, c_2, \ldots, c_{\min\{\xi, q(v)-2-|Z_F|\}}\} \cup Z_F$.
end if
Output $\mu_P^{<}(v;\Pi)$, and halt.

The following lemma follows from the above argument.

Lemma 4.4. *We can compute $\mu_P^{<}(v)$ by using Algorithm 3.*

The following lemma follows from Lemmas 4.2 and 4.4.

Lemma 4.5. *We can compute $\mu_P(v)$ by using $\mu_X(c)$ for vertices c in C_v and symbols X in $\{P, \overline{P}, F\}$.*

4.3. Algorithm for computing $\mu_{\overline{P}}(v)$ and $\mu_F(v)$

Here we consider how to compute $\mu_{\overline{P}}(v)$. By using Lemma 3.7, we can compute $\mu_{\overline{P}}(v)$ in the similar way as used in the case of computing $\mu_P(v)$. We define $\mu_{\overline{P}}^{=}(v)$ and $\mu_{\overline{P}}^{<}(v)$ by

$$\mu_{\overline{P}}^{=}(v) := \begin{cases} \max\{|M| \mid M \in \mathbf{M}_{\overline{P}}^{=}(v)\} & \text{if } \mathbf{M}_{\overline{P}}^{=}(v) \neq \emptyset \\ -\infty & \text{if } \mathbf{M}_{\overline{P}}^{=}(v) = \emptyset, \end{cases}$$

$$\mu_{\overline{P}}^{<}(v) := \begin{cases} \max\{|M| \mid M \in \mathbf{M}_{\overline{P}}^{<}(v)\} & \text{if } \mathbf{M}_{\overline{P}}^{<}(v) \neq \emptyset \\ -\infty & \text{if } \mathbf{M}_{\overline{P}}^{<}(v) = \emptyset. \end{cases}$$

Clearly, $\mu_{\overline{P}}(v) = \max\{\mu_{\overline{P}}^{=}(v), \mu_{\overline{P}}^{<}(v)\}$.

We consider how to compute $\mu_{\overline{P}}^{=}(v)$. For each $i = 1, 2, \ldots, \ell(v)$, we define $\mu_{\overline{P}}^{=}(v, i)$ as the maximum-size of a member M in $\mathbf{M}_{\overline{P}}^{=}(v)$ such that

- $M \cap S_c \in \mathbf{M}_P(c) \cup \mathbf{M}_F(c)$ for every vertex c in K_v^{i-1},
- $M \cap S_c \in \mathbf{M}_P(c) \cup \mathbf{M}_{\overline{P}}(c)$ for every vertex c in C_v^i, and
- $M \cap S_c \in \mathbf{M}_{\overline{P}}(c)$ for every vertex c in $C_v \setminus K_v^i$.

If there is no such a member in $\mathbf{M}_{\overline{P}}^{=}(v)$, then we define $\mu_{\overline{P}}^{=}(v,i) := -\infty$. Lemma 3.5 implies that we have

$$\mu_{\overline{P}}^{=}(v) = \max\{\mu_{\overline{P}}^{=}(v,i) \mid i = 1, 2, \dots, \ell(v)\}.$$

Let i be an integer in $\{1, 2, \dots, \ell(v)\}$, and we consider how to compute $\mu_{\overline{P}}^{=}(v,i)$. Lemma 3.5 implies the following lemma.

Lemma 4.6. *We have $\mu_{\overline{P}}^{=}(v,i) \neq -\infty$ if and only if*

- *there is no vertex c in K_v^{i-1} such that $\mu_P(c) = \mu_F(c) = -\infty$,*
- *there are at most $q(v)$ vertices c in K_v^{i-1} such that $\mu_F(c) = -\infty$, and*
- *there are at least $q(v)$ vertices c in K_v^i such that $\mu_P(c) \neq -\infty$.*

Lemma 4.6 implies that we can decide whether $\mu_{\overline{P}}^{=}(v,i) \neq -\infty$ by using $\mu_X(c)$ for vertices c in C_v and symbols X in $\{P, \overline{P}, F\}$. Thus, in what follows, we assume that $\mu_{\overline{P}}^{=}(v,i) \neq -\infty$. Then, in the similar way as used in the case of computing $\mu_P^{=}(v,i)$, we can prove that $\mu_{\overline{P}}^{=}(v,i)$ can be computed by Algorithm 4.

Algorithm 4 Algorithm for computing $\mu_{\overline{P}}^{=}(v,i)$

1: Sort vertices in $Z_P^i \setminus Z_F^{i-1}$ as $c_1, c_2, \dots$ in such a way that $\varphi_{X_{c_1}^i}(c_1) \geq \varphi_{X_{c_2}^i}(c_2) \geq \cdots$.
2: Set $\Pi := \{c_1, c_2, \dots, c_{q(v)-|Z_F^{i-1}|}\} \cup Z_F^{i-1}$.
3: Compute $\mu_{\overline{P}}^{=}(v,i)$ by

$$\mu_{\overline{P}}^{=}(v,i) := \sum_{c \in \Pi} \mu_P(c) + \sum_{c \in C_v \setminus \Pi} \mu_{X_c^i}(c).$$

4: Output $\mu_{\overline{P}}^{=}(v,i)$, and halt.

Lemma 4.7. *We can compute $\mu_{\overline{P}}^{=}(v,i)$ by using Algorithm 4.*

Next we consider how to compute $\mu_{\overline{P}}^{\leq}(v)$. Lemma 3.6 implies the following lemma.

Lemma 4.8. *We have $\mu_{\overline{P}}^{\leq}(v) \neq -\infty$ if and only if*

- *there is no vertex c in C_v such that $\mu_P(c) = \mu_F(c) = -\infty$, and*
- *there are at most $q(v) - 1$ vertices in C_v such that $\mu_F(c) = -\infty$.*

Lemma 4.8 implies that we can decide whether $\mu_{\overline{P}}^{\leq}(v) \neq -\infty$ by using $\mu_X(c)$ for vertices c in C_v and symbols X in $\{P, \overline{P}, F\}$. Thus, in what follows, we assume that $\mu_{\overline{P}}^{\leq}(v) \neq -\infty$. In the similar way as used in the case of computing $\mu_P^{\leq}(v)$, we can prove that $\mu_{\overline{P}}^{\leq}(v)$ can be computed by Algorithm 5.

Lemma 4.9. *We can compute $\mu_{\overline{P}}^{\leq}(v)$ by using Algorithm 5.*

The following lemma follows from Lemmas 4.7 and 4.9.

Lemma 4.10. *We can compute $\mu_{\overline{P}}(v)$ by using $\mu_X(c)$ for vertices c in C_v and symbols X in $\{P, \overline{P}, F\}$.*

Lastly, we consider how to compute $\mu_F(v)$. If $t^-(v) = 0$, then $\mathbf{M}_F(v) = \emptyset$. If $t^-(v) > 0$, then Lemma 3.7 implies that

$$\mu_F(v) = \max\{\mu_{\overline{P}}^{=}(v,i) \mid i = 1, 2, \dots, t^-(v)\}.$$

Thus, we can compute $\mu_F(v)$ by using Algorithm 4.

Algorithm 5 Algorithm for computing $\mu^{\leq}_{\overline{P}}(v)$

Sort vertices in $C_v \setminus Z_F$ as $c_1, c_2, \ldots$ in such a way that $\varphi_F(c_1) \geq \varphi_F(c_2) \geq \cdots$.
if $\varphi_F(c_1) \leq 0$ **then**
  Set $\Pi := \emptyset$.
else
  Set ξ to be the maximum integer j in $\{1, 2, \ldots, |C_v \setminus Z_F|\}$ such that $\varphi_F(c_j) > 0$.
  Set $\Pi := \{c_1, c_2, \ldots, c_{\min\{\xi, q(v)-1-|Z_F|\}}\} \cup Z_F$.
end if
Compute $\mu^{\leq}_{\overline{P}}(v)$ by

$$\mu^{\leq}_{\overline{P}}(v) := \sum_{c \in \Pi} \mu_P(c) + \sum_{c \in C_v \setminus \Pi} \mu_F(c).$$

Output $\mu^{\leq}_{\overline{P}}(v)$, and halt.

Lemma 4.11. *We can compute $\mu_F(v)$ by using $\mu_X(c)$ for vertices c in C_v and symbols X in $\{P, \overline{P}, F\}$.*

The following theorem follows from Lemmas 4.5, 4.10, and 4.11.

Theorem 4.1. *Algorithm 1 can compute the optimal objective value of* MMSMT *in trees.*

Here we evaluate the time complexity of Algorithm 1. Define $n := |V|$ and $n_v := |C_v|$ for each vertex v in U. Assume that for every vertex v in V and every pair of vertices u, w in $N(v)$, we can decide whether $u \succsim_v w$ in $O(1)$ time. It is not difficult to see that we can compute C^i_v for all vertices v in U and all $i = 1, 2, \ldots, \ell(v)$ in $O(n \log n)$ time in the same way as sorting. Let v be a vertex in U, and assume that we know $\mu_X(c)$ for all vertices c in C_v and all symbols X in $\{P, \overline{P}, F\}$. Under this assumption, we evaluate the time required to compute $\mu_X(v)$ for each symbol X in $\{P, \overline{P}, F\}$. Since $\mu^{\leq}_{P}(v)$ and $\mu^{\leq}_{\overline{P}}(v)$ can be computed in $O(n_v \log n_v)$ time, we consider the time required to compute $\mu^{=}_{P}(v)$, $\mu^{=}_{\overline{P}}(v)$, and $\mu_F(v)$. Since we can compute $\mu^{=}_{\overline{P}}(v)$ and $\mu_F(v)$ in the similar way, we concentrate on the time required to compute $\mu^{=}_{P}(v)$. For each $i = t^+(v), t^+(v)+1, \ldots, \ell(v)+1$, we can decide whether $\mu^{=}_{P}(v, i) \neq -\infty$ in $O(n_v)$ time by using Lemma 4.1. Furthermore, for each $i = t^+(v), t^+(v)+1, \ldots, \ell(v)+1$ such that $\mu^{=}_{P}(v, i) \neq -\infty$, we can compute $\mu^{=}_{P}(v, i)$ in $O(n_v \log n_v)$ time by Algorithm 2. Since $\ell(v) \leq n_v$, we can compute $\mu^{=}_{P}(v)$ in $O(n_v^2 \log n_v)$ time. Thus, if we naively implement Algorithm 1, then its time complexity is $O(n^2 \log n)$. In the next section, we prove that Algorithm 1 can be implemented in $O(n \log n)$ time with a more sophisticated data structure.

5. Faster Implementation

The goal of this section is to prove that Algorithm 1 can be implemented in $O(n \log n)$ time, where n is the number of vertices of the input tree. For achieving this time complexity, we use a binary (min) heap that is a binary rooted tree $\mathbf{H}$ such that each vertex of $\mathbf{H}$ corresponds to some vertex of T, and each vertex of $\mathbf{H}$ is associated with a value, called a *key*. We do not distinguish between a vertex h of $\mathbf{H}$ and the vertex of T corresponding to h. For each non-root vertex h of $\mathbf{H}$, the key of h is more than or equal to that of its parent, which implies that the key of the root of $\mathbf{H}$ is minimum among all vertices of $\mathbf{H}$. If the number of vertices of $\mathbf{H}$ is m and we can directly access to a vertex of $\mathbf{H}$ by using a pointer, then we can delete a vertex of $\mathbf{H}$, insert a new vertex to $\mathbf{H}$, and change the key of some vertex of $\mathbf{H}$ in $O(\log m)$ time, respectively. See, e.g., [6] for details of a binary heap. We denote by $\mathsf{r}(\mathbf{H})$ and $\mathsf{V}(\mathbf{H})$ the root of $\mathbf{H}$ and the set of vertices in $\mathbf{H}$, respectively. For each vertex v in $\mathsf{V}(\mathbf{H})$, we denote by $\mathsf{key}(v)$ the key of v.

From now on, we give a faster implementation. Let v be a vertex in U. Since it is not difficult to see that we can compute $\mu_P^{=}(v)$ and $\mu_F(v)$ in the similar way, we concentrate on the time required to compute $\mu_P^{=}(v)$. In what follows, we use the notation introduced in Section 4.1.

5.1. Step 1

We first compute the set I of integers in $\{t^+(v), t^+(v)+1, \ldots, \ell(v)+1\}$ such that $\mu_P^{=}(v,i) \neq -\infty$ by using Algorithm 6.

Algorithm 6 Algorithm for computing I

Set $I := \emptyset$, $i := 1$, $\Delta_P^0 := 0$, and $\Delta_F^0 := 0$.
while $i \leq \ell(v)+1$ **do**
  if $i > 1$ **then**
    if there is a vertex c in C_v^{i-1} such that $\mu_P(c) = \mu_F(c) = -\infty$ **then**
      Output I, and halt.
    end if
    Set $\Delta_F^{i-1} := \Delta_F^{i-2} + |\{c \in C_v^{i-1} \mid \mu_F(c) = -\infty\}|$.
    if $\Delta_F^{i-1} > q(v) - 1$ **then**
      Output I, and halt.
    end if
  end if
  Set $\Delta_P^i := \Delta_P^{i-1} + |\{c \in C_v^i \mid \mu_P(c) \neq -\infty\}|$.
  if $i \geq t^+(v)$ and $\Delta_P^i \geq q(v) - 1$ **then**
    Set $I := I \cup \{i\}$.
  end if
  Set $i := i + 1$.
end while
Output I, and halt.

It is not difficult to see that in Algorithm 6, we have $\Delta_P^i = |Z_P^i|$ and $\Delta_F^i = |Z_F^i|$. Thus, the correctness of Algorithm 6 immediately follows from Lemma 4.1. The time complexity of Algorithm 6 is $O(n_v)$. It is not difficult to see that I consists of consecutive integers. Thus, in what follows, we assume that $I = \{\mathsf{L}, \mathsf{L}+1, \ldots, \mathsf{R}\}$.

5.2. Step 2

The next step is to compute $\mu_P^{=}(v, \mathsf{L})$ and construct a binary heap $\mathbf{H}$ used later. We compute $\mu_P^{=}(v, \mathsf{L})$ and construct a binary heap $\mathbf{H}$ by using Algorithm 7.

Here we prove the correctness of Algorithm 7, i.e., we prove that $\mu_P^{=}(v, \mathsf{L}) = \zeta(\mathbf{H})$ when Algorithm 7 halts. It is not difficult to see that during this algorithm, we have

$$\zeta(\mathbf{H}) = \sum_{c \in \mathsf{V}(\mathbf{H}) \cup Z_F^{\mathsf{L}-1}} \mu_P(c) + \sum_{c \in C_v \setminus (\mathsf{V}(\mathbf{H}) \cup Z_F^{\mathsf{L}-1})} \mu_{X_c^{\mathsf{L}}}(c) + 1.$$

The definition of L implies that $\Delta_F^{\mathsf{L}-1} \leq q(v) - 1$. Thus, when Algorithm 7 halts, we have $\Delta = q(v) - 1$, i.e., $|\mathsf{V}(\mathbf{H}) \cup Z_F^{\mathsf{L}-1}| = q(v) - 1$. Thus, $\mathsf{V}(\mathbf{H}) \cup Z_F^{\mathsf{L}-1}$ is a $(P, =, \mathsf{L})$-feasible subset of K_v^{L}. Furthermore, if we set $\Pi := \mathsf{V}(\mathbf{H}) \cup Z_F^{\mathsf{L}-1}$, then the definition of a binary (min) heap implies that Π maximizes the first term in the last line of (4.2) in the case of $i = \mathsf{L}$. This completes the correctness of Algorithm 7. The time complexity of Algorithm 7 is clearly $O(n_v \log n_v)$.

Algorithm 7 Algorithm for computing $\mu^{=}_{\overline{P}}(v, \mathsf{L})$ and constructing $\mathbf{H}$

Set $\Delta := \Delta_F^{\mathsf{L}-1}$ and $\mathbf{H}$ to be an empty binary heap.
Set $\zeta(\mathbf{H})$ by

$$\zeta(\mathbf{H}) := \sum_{c \in Z_F^{\mathsf{L}-1}} \mu_P(c) + \sum_{c \in C_v \setminus Z_F^{\mathsf{L}-1}} \mu_{X_c^{\mathsf{L}}}(c) + 1.$$

for all vertices c in $(K_v^{\mathsf{L}-1} \setminus Z_F^{\mathsf{L}-1}) \cap Z_P^{\mathsf{L}}$ **do**
  if $\Delta < q(v) - 1$ **then**
    Insert c with a key $\varphi_F(c)$ to $\mathbf{H}$, and set $\zeta(\mathbf{H}) := \zeta(\mathbf{H}) + \varphi_F(c)$ and $\Delta := \Delta + 1$.
  end if
  if $\Delta = q(v) - 1$, $\mathsf{V}(\mathbf{H}) \neq \emptyset$, and $\varphi_F(c) > \mathsf{key}(\mathsf{r}(\mathbf{H}))$ **then**
    Remove $\mathsf{r}(\mathbf{H})$ from $\mathbf{H}$, and insert c with a key $\varphi_F(c)$ to $\mathbf{H}$.
    Set $\zeta(\mathbf{H}) := \zeta(\mathbf{H}) - \mathsf{key}(\mathsf{r}(\mathbf{H})) + \varphi_F(c)$.
  end if
end for
for all vertices c in $C_v^{\mathsf{L}} \cap Z_P^{\mathsf{L}}$ **do**
  if $\Delta < q(v) - 1$ **then**
    Insert c with a key $\varphi_{\overline{P}}(c)$ to $\mathbf{H}$, and set $\zeta(\mathbf{H}) := \zeta(\mathbf{H}) + \varphi_{\overline{P}}(c)$ and $\Delta := \Delta + 1$.
  end if
  if $\Delta = q(v) - 1$, $\mathsf{V}(\mathbf{H}) \neq \emptyset$, and $\varphi_{\overline{P}}(c) > \mathsf{key}(\mathsf{r}(\mathbf{H}))$ **then**
    Remove $\mathsf{r}(\mathbf{H})$ from $\mathbf{H}$, and insert c with a key $\varphi_{\overline{P}}(c)$ to $\mathbf{H}$.
    Set $\zeta(\mathbf{H}) := \zeta(\mathbf{H}) - \mathsf{key}(\mathsf{r}(\mathbf{H})) + \varphi_{\overline{P}}(c)$.
  end if
end for
Output $\zeta(\mathbf{H})$ and $\mathbf{H}$. Then, halt.

5.3. Step 3

Lastly, we consider how to compute $\mu^{=}_{\overline{P}}(v, i)$ for each $i = \mathsf{L}+1, \mathsf{L}+2, \ldots, \mathsf{R}$. We can compute these values by using Algorithm 8.

For each $i = \mathsf{L}, \mathsf{L}+1, \ldots, \mathsf{R}-1$, let $\widehat{\mu}^{=}_{\overline{P}}(v, i)$ be the maximum size of a member M in $\mathbf{M}^{=}_{\overline{P}}(v, i)$ such that

- $M \cap S_c \in \mathbf{M}_P(c) \cup \mathbf{M}_F(c)$ for every vertex c in K_v^i, and
- $M \cap S_c \in \mathbf{M}_{\overline{P}}(c)$ for every vertex c in $C_v \setminus K_v^i$.

In the ith iteration of Algorithm 8, we have $\mu^{=}_{\overline{P}}(v, i-1)$ and the corresponding binary heap $\mathbf{H}$ as an initial input. For computing $\mu^{=}_{\overline{P}}(v, i)$ from these inputs, we first compute $\widehat{\mu}^{=}_{\overline{P}}(v, i-1)$ (in the lines 3 to 22). Notice that for every vertex c in C_v such that $\mu_P(c) \neq -\infty$ and $\mu_F(c) \neq -\infty$, we have $\varphi_F(c) \geq \varphi_{\overline{P}}(c)$. Furthermore, since $X_c^{i-1} = \overline{P}$ and $X_c^i = F$ for every vertex c in C_v^{i-1}, we need to do the operation in the line 7. For the same reason, we need to do the operation in the line 19. In addition, since $X_c^{i-1} = \overline{P}$ for every vertex c in C_v^{i-1}, we add $\varphi_{\overline{P}}(c)$ to $\zeta(\mathbf{H})$ in the lines 12 and 16. However, since $X_c^i = F$ for every vertex c in C_v^{i-1}, we set the key of c to be $\varphi_F(c)$ in the line 15. In the lines 23 to 28, we compute $\mu^{=}_{\overline{P}}(v, i)$. During these lines, we have

$$\zeta(\mathbf{H}) = \sum_{c \in \mathsf{V}(\mathbf{H}) \cup Z_F^{i-1}} \mu_P(c) + \sum_{c \in C_v \setminus (\mathsf{V}(\mathbf{H}) \cup Z_F^{i-1})} \mu_{X_c^i}(c) + 1.$$

Thus, we can compute $\mu^{=}_{\overline{P}}(v, i)$ in the ith iteration. The time complexity of Algorithm 8 is clearly $O(n_v \log n_v)$.

Algorithm 8 Algorithm for computing $\mu^{=}_{\overline{P}}(v,i)$ for each $i = \mathsf{L}+1, \mathsf{L}+2, \ldots, \mathsf{R}$

Set $i := \mathsf{L}+1$, $\mathbf{H} :=$ the binary heap obtained by Algorithm 7, and $\zeta(\mathbf{H}) := \mu^{=}_{\overline{P}}(v, \mathsf{L})$.
while $i \leq \mathsf{R}$ **do**
  for all vertices c in $C_v^{i-1} \cap \mathsf{V}(\mathbf{H})$ **do**
    if $\mu_F(c) = -\infty$ **then**
      Remove c from $\mathbf{H}$.
    else
      Change the key of c from $\varphi_{\overline{P}}(c)$ to $\varphi_F(c)$.
    end if
  end for
  for all vertices c in $C_v^{i-1} \setminus \mathsf{V}(\mathbf{H})$ **do**
    if $\mu_F(c) = -\infty$ **then**
      Remove $\mathsf{r}(\mathbf{H})$ from $\mathbf{H}$, and set $\zeta(\mathbf{H}) := \zeta(\mathbf{H}) - \mathsf{key}(\mathsf{r}(\mathbf{H})) + \varphi_{\overline{P}}(c)$.
    else
      if $\mathsf{V}(\mathbf{H}) \neq \emptyset$ and $\varphi_F(c) > \mathsf{key}(\mathsf{r}(\mathbf{H}))$ **then**
        Remove $\mathsf{r}(\mathbf{H})$ from $\mathbf{H}$, and insert c with a key $\varphi_F(c)$ to $\mathbf{H}$.
        Set $\zeta(\mathbf{H}) := \zeta(\mathbf{H}) - \mathsf{key}(\mathsf{r}(\mathbf{H})) + \varphi_{\overline{P}}(c)$.
      end if
      if $\mathsf{V}(\mathbf{H}) = \emptyset$ and/or $\varphi_F(c) \leq \mathsf{key}(\mathsf{r}(\mathbf{H}))$ **then**
        Set $\zeta(\mathbf{H}) := \zeta(\mathbf{H}) - \mu_{\overline{P}}(c) + \mu_F(c)$.
      end if
    end if
  end for
  for all vertices c in $C_v^i \cap Z_P^i$ **do**
    if $\varphi_{\overline{P}}(c) > \mathsf{key}(\mathsf{r}(\mathbf{H}))$ **then**
      Remove $\mathsf{r}(\mathbf{H})$ from $\mathbf{H}$, and insert c with a key $\varphi_{\overline{P}}(c)$ to $\mathbf{H}$.
      Set $\zeta(\mathbf{H}) := \zeta(\mathbf{H}) - \mathsf{key}(\mathsf{r}(\mathbf{H})) + \varphi_{\overline{P}}(c)$.
    end if
  end for
  Output $\zeta(\mathbf{H})$ (for $\mu^{=}_{\overline{P}}(v,i)$), and set $i := i+1$.
end while
Halt.

In the similar way, we can compute $\mu^{=}_{\overline{P}}(v)$ and $\mu_F(v)$ in $O(n_v \log n_v)$ time. This completes the proof of the fact that Algorithm 1 can be implemented in $O(n \log n)$ time.

5.4. Finding an optimal solution

Here we consider how to find an optimal solution. For this, it suffices to compute $\partial_v M$ for some maximum-size member M in $\mathbf{M}^{=}_{\overline{P}}(v)$. For computing this, we first compute an integer ξ in $\{\mathsf{L}, \mathsf{L}+1, \ldots, \mathsf{R}\}$ such that $\mu^{=}_{\overline{P}}(v) = \mu^{=}_{\overline{P}}(v, \xi)$. If $\xi = \mathsf{L}$, then we run Algorithm 7, and output $\mathsf{V}(\mathbf{H}) \cup Z_F^{\mathsf{L}-1}$. If $\xi > \mathsf{L}$, then we run Algorithms 7 and 8. Then, we stop Algorithm 8 when the ξth iteration terminates, and output $\mathsf{V}(\mathbf{H}) \cup Z_F^{\xi-1}$. We can treat $\mathbf{M}^{=}_{\overline{P}}(v)$ and $\mathbf{M}_F(v)$ in the similar way. By using this algorithm from the root r, we can find an optimal solution in $O(n \log n)$ time.

Acknowledgements. Naoyuki Kamiyama was supported by JST, PRESTO. The authors would like to thank anonymous referees for helpful comments on an earlier version of this paper.

References

[1] D. Gale and L.S. Shapley: College admissions and the stability of marriage. *The American Mathematical Monthly*, **69-1** (1962), 9–15.

[2] D. Gale and M. Sotomayor: Some remarks on the stable matching problem. *Discrete Applied Mathematics*, **11-3** (1985), 223–232.

[3] R.W. Irving: Stable marriage and indifference. *Discrete Applied Mathematics*, **48-3** (1994), 261–272.

[4] R.W. Irving, D.F. Manlove, and G. O'Malley: Stable marriage with ties and bounded length preference lists. *Journal of Discrete Algorithms*, **7-2** (2009), 213–219.

[5] Z. Király: Linear time local approximation algorithm for maximum stable marriage. *Algorithms*, **6-3** (2013), 471–484.

[6] J. Kleinberg and É. Tardos: *Algorithm Design* (Addison-Wesley, Boston, 2005).

[7] D.F. Manlove: *Algorithmics of Matching Under Preferences* (World Scientific, Singapore, 2013).

[8] D.F. Manlove, R.W. Irving, K. Iwama, S. Miyazaki, and Y. Morita: Hard variants of stable marriage. *Theoretical Computer Science*, **276-1&2** (2002), 261–279.

[9] E. McDermid: A 3/2-approximation algorithm for general stable marriage. In S. Albers, A. Marchetti-Spaccamela, Y. Matias, S. Nikoletseas, and W. Thomas (eds.): *Proceedings of the 36th International Colloquium on Automata, Languages and Programming Automata, Languages and Programming, Part I*, volume 5555 of *Lecture Notes in Computer Science* (Springer-Verlag, Berlin, Heidelberg, 2009), 689–700.

[10] K. Paluch: Faster and simpler approximation of stable matchings. *Algorithms*, **7-2** (2014), 189–202.

[11] S. Tayu and S. Ueno: Stable matchings in trees. IPSJ SIG Technical Report, Vol.2013-AL-145, No.10 (2013).

Naoyuki Kamiyama
Kyushu University
744 Motooka, Nishi-ku
Fukuoka 819-0395, Japan
E-mail: kamiyama@imi.kyushu-u.ac.jp