Many-to-many perfect matching

MUSASHI TAKANEZAWA and YOSHIFUMI MANABE, Faculty of Informatics, Kogakuin University, Japan

This paper considers a new type of two-sided matching in which multiple numbers of agents are perfectly matched on both sides. Such matching can be used between multiple major students and laboratories. The many-to-many perfect matching problem cannot be solved by existing many-to-many matching algorithms, since the perfect property, which is a global property, cannot be represented by the participants' preferences, which are local properties. This paper gives a DA(Deferred Acceptance) mechanism to match each student to the given number of different laboratories without a blocking pair by introducing a master list of students to resolve ties between students.

CCS Concepts: • **Mathematics of computing** → **Combinatorial algorithms**.

Additional Key Words and Phrases: many to many matching, perfect matching, DA mechanism, stable matching

ACM Reference Format:
Musashi Takanezawa and Yoshifumi Manabe. 2022. Many-to-many perfect matching. In . ACM, New York, NY, USA, 11 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

This paper considers a new type of two-sided matching in which multiple numbers of agents are perfectly matched on both sides. Such matching can be used between multiple major students and laboratories[1]. The students have multiple majors, thus each student s_i must be matched to k_i different laboratories. Each laboratory l_i has c_i slots to accept students. The total number of slots of laboratories equals the total number of applications by students. The matching result must be perfect, that is, there must not be a student who is accepted in less than k_i different laboratories. It means that there must not be a laboratory that accepts less than c_i different students. Such a perfect matching problem cannot be solved by existing many-to-many matching algorithms, since the perfect property, which is a global property, cannot be represented by the participants' preferences, which are local properties. This paper gives a DA(Deferred Acceptance) mechanism to match each student to k_i different laboratories without a blocking pair by introducing a master list of students to resolve ties between students.

The matching problem has been discussed for many years[18]. One-to-one matching was first discussed and then many-to-one and many-to-many matchings were considered. DA mechanism was proposed to solve one-to-one matching problem[7].

[1]In the Faculty of Informatics, Kogakuin University, undergraduate students in the 3rd grade must have seminars in two different laboratories.

Manuscript submitted to ACM

Many algorithms have been proposed for many-to-one matching problem[6][8][23]. Such a problem is used for matching between interns and hospitals or students and laboratories, in which each intern or student is matched to one hospital or laboratory. Each hospital or laboratory accepts many interns or students. DA-like mechanisms to solve these problems are shown. A master list was introduced to solve a matching with a minimum quota[6].

There are several works that considers many-to-many matching[1, 4, 5, 11, 13–17, 19, 21, 22]. These works mainly consider a matching between workers and farms. The preference of each worker is a sequence of a subset of firms, that is, the number of the match can be arbitrary. These works do not consider the perfect property and these algorithms cannot be used for this paper's problem. In [2], many-to-many matching between students and courses was discussed. However, each course has no preference among students, thus the algorithm cannot be used for this paper's problem. In [3], many-to-many matching between students and courses with each course's preference was discussed, although the perfect property was not considered. Many-to-many matching when the preference has max-min property was shown in [9, 12]. The max-min preference differs from the preference defined in this paper.

Another type of many-to-many matching is the case when the agents are not divided into disjoint sets (like students and laboratories). Matching for such cases were considered[10, 20].

In many-to-one matching problems, the perfect property can be introduced by setting all laboratories to accept up to the number of the laboratory's capacity. However, as shown in section 2, the perfect property in a many-to-many matching problem cannot be represented by each laboratory/student's local preference. A naive algorithm might result in a matching in which a student is matched to a laboratory multiple times. Avoiding such infeasible matching is necessary to obtain a perfect many-to-many matching. This paper shows a new DA mechanism to solve the many-to-many perfect matching. The algorithm uses a master list to resolve ties between students. Section 2 shows the definition of the problem. Section 3 shows the matching algorithm and its correctness proof. Section 4 concludes the paper.

2 PROBLEM DEFINITION

$S = \{s_1, s_2, \ldots, s_m\}$ is the set of students. $m = |S|$ is the number of students. $L = \{l_1, l_2, \ldots, l_n\}$ is the set of laboratories. $n = |L|$ is the number of laboratories. Each student has a preference over L. $l \succ_{s_i} l'$ means student s_i prefers l than l'. The preference of each student has no ties, thus all laboratories are strictly ranked by each student.

Each laboratory has a preference over S. $s \succ_{l_i} s'$ means laboratory l_i prefers s than s'. The preference of each laboratory has no ties, thus all students are strictly ranked by each laboratory.

Each laboratory $l_i (1 \leq i \leq n)$ has its capacity $c_i (1 \leq i \leq n)$. $0 < c_i \leq m (1 \leq i \leq n)$ must be satisfied. If $c_i > m$, no feasible matching exists.

Each student $s_i (1 \leq i \leq m)$ has its applies $k_i (1 \leq i \leq m)$. $0 < k_i \leq n (1 \leq i \leq m)$ must be satisfied. If $k_i > n$, no feasible matching exists.

The number of slots must satisfy the following equation: $\sum_{i=1}^{n} c_i = \sum_{i=1}^{m} k_i$. Thus, there are no extra slots in each laboratory.

The student and laboratory's preferences are defined for a single element. Since a many-to-many matching is executed, we need to define each student and laboratory's preference over multiple assignment results. This paper assumes the following simple preference for sets of elements. Consider student s_i's two matching

result $\alpha = (l_{a_1}, l_{a_2}, \ldots, l_{a_{k_i}})$ and $\beta = (l_{a'_1}, l_{a'_2}, \ldots, l_{a'_{k_i}})$, in which the laboratories are sorted as $l_{a_j} \succ_{s_i} l_{a_{j+1}}$ and $l_{a'_j} \succ_s l_{a'_{j+1}} (1 \leq j \leq k_i - 1)$. The two results satisfy $\alpha \succ_{s_i} \beta$ if and only if there is some $d (1 \leq d \leq k_i)$ that satisfies $l_{a_j} = l_{a'_j}$ for all $j < d$ and $l_{a_d} \succ_{s_i} l_{a'_d}$, that is the same as the lexicographic order ,which is the same as the one in [2]. Each laboratory has the same preference for a set of students. Note that in the many-to-one matching in [6], this preference for sets of students or interns is implicitly assumed as the laboratory or hospital's preference.

We introduce a master list ML to break ties between students. ML is a sequence of S's elements, for example, the order decided by the grades of examinations. The usage of ML is shown later. Note that without loss of generality, the index of S is changed so that $[s_1, s_2, \ldots, s_m]$ is ML.

A many-to-many perfect matching between S and L is $\mu : S \cup T \to 2^{S \cup L}$ that satisfies the following properties.

(1) $\mu(s_i) \subseteq L$ and $\mu(l_i) \subseteq S$.
(2) $s_i \in \mu(l_j)$ if and only if $l_j \in \mu(s_i)$.
(3) $|\mu(s_i)| = k_i$ for any $s_i \in S$.
(4) $|\mu(l_i)| = c_i$ for any $l_i \in L$.

A matching that satisfies the first and the second property but does not satisfy the third or the fourth property is called an infeasible matching. A matching that satisfies all the conditions is called feasible.

Note that there are problem instances that have no feasible matching. Such a problem is called an infeasible problem instance. An example of an infeasible problem instance is shown below.

(Example 1) $m = 6, n = 4$, $c_1 = c_2 = 5, c_3 = c_4 = 1$, $k_1 = k_2 = k_3 = 3$, $k_4 = k_5 = k_6 = 1$.

The procedure to check feasibility is shown below. A problem instance is feasible if and only if the return value $va = 0$. In an infeasible matching, a student is assigned to multiple slots in the same laboratory. Thus, a matching that avoids an infeasible result is obtained by a procedure that the student with the larger number of applies and the laboratory with the larger number of remaining slots are matched.

Algorithm 1 Feasibility check procedure

```
1: procedure FEASIBLE(L) /* The students are renamed by the decreasing order of k_i. */
2:     va = 0 /* the number of unassigned slots */
3:     for i = 1 to m do
4:         Let x be the number of laboratories with at least one remaining slot.
5:         y = min(x, k_i)
6:         va = va + (k_i − y)
7:         Delete one slot for each of y laboratories with the larger number of remaining slots.
8:     end for
9:     return(va) /* the number of unassigned slots */
10: end procedure
```

Let us execute the feasibility check procedure for Example 1. Since $k_1 = 3$, one slot of l_1, l_2, and l_3 are removed. Since $k_2 = 3$, one slot of l_1, l_2, and l_4 are removed. At this instant, the remaining slots of l_1 and l_2 are 3. The remaining slots of l_3 and l_4 are 0. Thus, it is impossible to assign s_3 with $k_3 = 3$. va becomes 1. The remaining students s_4, s_5, and s_6 can be assigned thus the return value $va = 1$. This problem instance is detected as infeasible.

In the rest of the paper, we assume that the given problem instance is feasible.

A many-to-many perfect matching μ is stable if there is no pair of student and laboratory (s, l) that satisfy the following conditions.

- $s \notin \mu(l)$.
- There is a pair (s', l') that satisfy the following conditions: $s \succ_l s'$, $l \succ_s l'$, $s' \in \mu(l)$, $s \in \mu(l')$, and swapping the matching from (s, l') and (s', l) to (s, l) and (s', l') results in a feasible matching.

(s, l) is called a blocking pair. We call (s', l') a supporting pair of the blocking pair. The condition means that s and l prefer each other over the currently matched one. The difference between the definition of a blocking pair in the usual one-to-one matching [7] is that the new matching result must be feasible.

3 MATCHING ALGORITHM USING MASTER LIST

DA mechanism obtains a matching that has no blocking pair for one-to-one matching and many-to-one matching, since each person applies to the best one and each person in the opposite side accepts the best one(s). We can consider the following standard DA mechanism for many-to-many matching by generating k_i agents for each student, but the mechanism does not work well, as shown below.

Algorithm 2 Standard DA mechanism that does not work well

1: Each student s_i makes k_i agents $s_{i,0}, s_{i,1}, \dots s_{i,k_i-1}$. Agent $s_{i,x}$ applies to s_i's $(x+1)$-th preferred laboratory.
2: The algorithm terminates if the number of applicants in each laboratory l_i is c_i.
3: Each laboratory l_j tentatively accepts its most preferred set of agents up to c_i. l_j rejects the rest of the agents.
4: Each rejected agent applies to the most preferred laboratory that no agent of the same student has applied to.
5: Goto step 2.

(Example 2) $n = 4$, $m = 4$, $c_1 = c_2 = c_3 = c_4 = 3$, $k_1 = k_2 = k_3 = k_4 = 3$, Every student's preference: $l_1 \succ l_2 \succ l_3 \succ l_4$. Every laboratory's preference: $s_1 \succ s_2 \succ s_3 \succ s_4$.

Execute Algorithm 2 to this example. $s_{1,0}, s_{2,0}, s_{3,0}$, and $s_{4,0}$ apply to l_1. $s_{1,1}, s_{2,1}, s_{3,1}$, and $s_{4,1}$ apply to l_2. $s_{1,2}, s_{2,2}, s_{3,2}$, and $s_{4,2}$ apply to l_3. l_1 rejects $s_{4,0}$. l_2 rejects $s_{4,1}$. l_3 rejects $s_{4,2}$. $s_{4,0}$ applies to l_4. Now there are two unaccepted agents $s_{4,1}$ and $s_{4,2}$. These agents are called vacant agents. There are two vacant slots in l_4, but l_4 cannot accept these agents since s_4 are assigned multiple times to l_4 and it is not a feasible matching. Thus, the standard DA mechanism does not work well.

If we apply the T-algorithm for many-to-many matching in [4], the perfect property cannot be represented by each player's preference. The result is $\mu(s_1) = \mu(s_2) = \mu(s_3) = \{l_1, l_2, l_3\}$ and $\mu(s_4) = \{l_4\}$. Thus, the vacant slots remain and the matching is not perfect.

Some students must be accepted by a special rule to avoid infeasible results. We introduce a master list ML to solve the problem. ML is a sequence of students, for example, the students are ordered by the increasing order of scores of examinations. If it is impossible to assign all students using standard DA-mechanism, the assignment of some number of agents is restricted to make a feasible result. Such agents are called restricted agents. The other agents are called free agents. The restricted agents are decided using ML in the round-robin manner. For example, if $m = 4$, $ML = [s_1, s_2, s_3, s_4]$, $k_1 = 1$, $k_2 = k_3 = k_4 = 5$

and 6 restricted agents are needed, one agent of s_1, two agents of s_2 and s_3, and one agent of s_4 become the restricted agents. Since s_1 have one agent, it is impossible to have two restricted agents. Thus, another student in ML has more restricted agents instead of s_1. The restriction is set not for a student but for an agent, since restricting all agents of a student seems to be too punishing for the student.

The new algorithm is shown in Algorithms 3 and 4.

Algorithm 3 subroutine DA

procedure DA(L, L_v) /* L and L_v are sets of laboratories and $L_v \subset L$. */
  Each student s_i makes k_i agents $s_{i,0}, s_{i,1}, \ldots s_{i,k_i-1}$.
  Each free agent applies to the most preferred laboratory in L.
  Each restricted agent applies to the most preferred laboratory in L_v.
  repeat
    Each laboratory l_j whose applicants are more than its capacity c_j rejects the agents using l_j's preference so that the number of applicants becomes c_j.
    l_i (temporally) accepts all agents that are not rejected.
    Each rejected agent $s_{i.j}$ applies to the next most preferred laboratory.
    If an agent is rejected by all laboratories, the agent becomes vacant.
  until All agents become accepted or vacant.
  Let L' be laboratories that have vacant slots and $L' \cap L_v = \phi$. Let va be the number of vacant slots in L'.
  return(L', va)
end procedure

The outline of the procedure is as follows. First, execute a DA mechanism just like the one in Algorithm 2. The result might have vacant slots shown as in Example 2. Let the laboratories with vacant slots be L_v. Let $esum$ be the number of vacant slots. In the next round, $esum$ agents become restricted agents. They are forced to apply only to L_v to fill the vacant slots. The agents are selected using ML. Students are selected one by one from the top of ML in the round-robin manner. For the selected student, one agent is set to a restricted agent. However, there are cases when the restriction is not effective. The first case is the student has a vacant agent. Even if the vacant agent becomes a restricted agent, the agent does not fill the vacant slot. The second case is an agent that is accepted by L_v. The agent does not fill the vacant slot even if the agent becomes a restricted agent. In the other case, that is, an agent is accepted by $L - L_v$, the restriction might fill a vacant slot, thus the restriction is effective. The i-th restriction to student s is effective if the total numbers of s's vacant agents and s's agents accepted by L_v is less than $i-1$. If a restriction is effective, count up the variable e. The restrictions are added until e becomes $esum$. Note that the restriction might not always work well in the next round, since the newly available slots in $L - L_v$ might be filled by some agents other than the vacant agents and the vacant agents might be still vacant in the next round. Such a case is shown below in Example 2.

With the restricted preference list for some agents, the DA algorithm is executed again. The restricted agent applies only to L_v. Each laboratory's preference is changed as follows: for a free agent and a restricted agent of the same student, the laboratory prefers the restricted agent. Since the free agent might be accepted by some other laboratory in $L - L_v$, rejecting a free agent will result in a better result for the student.

The possible assignment results of the $z(> 1)$-th round are categorized into the following three cases. The first one is an enlargement of L_v. Some agents are forced to apply to L_v, thus the vacant agents are

Algorithm 4 Many-to-many perfect matching algorithm

```
1: procedure MANY-TO-MANY-MATCHING(L) /* L: set of laboratories. */
2:     Set all agents as free.
3:     L_v = φ;
4:     esum = 0;
5:     z = 1; /* Round 1. */
6:     repeat
7:         (L', va) = DA(L, φ);
8:         if z = 1 and L' = φ then terminate; /* matching is obtained */
9:         end if
10:        if va > 0 then
11:            L_v = L_v ∪ L';
12:        else/* va = 0 */
13:            va = feasible(L_v) ;
14:        end if
15:        if va > 0 then
16:            esum = esum + va;
17:            e = 0;
18:            repeat
19:                Select student s_i from ML one by one in the round-robin manner.
20:                One agent in s_i (if such an agent is available) becomes restricted. /* must apply to L_v */
21:                If the restriction is effective, set e = e + 1;
22:                /* The definition of effectiveness is shown in sentences below */
23:            until e = esum
24:            z = z + 1; /* next round */
25:        end if
26:    until va = 0 /* no enlargement of L_v and L_v is feasible */
27:    many − to − many − matching(L_v); /* execute for L_v */
28: end procedure
```

accepted by a laboratory $l \in L - L_v$, but there are new vacant slots in l. Such a case is shown below in Example 3. In the case, new vacant laboratory L_v becomes $L_v \cup \{l\}$. With the change, execute the same procedure again.

The second case is no enlargement of L_v, but the restriction is not enough. Execute a feasibility test for L_v with the agents currently accepted by L_v and vacant agents. If the current agents are not feasible for L_v, we need to force more agents to apply to L_v. Count the number of necessary agents that must be forced to apply to L_v. Note that in the making of the restriction, the condition to decide a restriction is effective or not differs from the first round, in which there is no restricted agent in the previous round. The decision algorithm from the second round is as follows. First, order the agents in the following manner: the top is restricted agents, the second is free vacant agents, and the third is free agents accepted by L_v, and the last is free agents accepted by $L - L_v$. When an agent is set as a restricted agent (note that a restricted agent in the previous round is set restricted again) by the following rule:

(1) Until the number of agents restricted in the previous round: the restriction is effective.

(2) Beyond (1), until the total numbers of vacant agents and free agents accepted by L_v: the restriction is not effective.

(3) Beyond (2) (agents accepted by $L - L_v$): the restriction is effective.

Note that this condition is optimistic and the restrictions in Case (1) might not be effective, since the restriction in the previous round might be set to an agent accepted by L_v. If the number of restrictions is not enough, the feasibility test fails in the next round and more restrictions are set. Such a case is shown in Example 3 below.

The last case is the feasibility test succeeds. The feasibility test is executed with L_v, the agents accepted by L_v, and vacant agents. In this case, execute the assignment again for L_v with the agents. The procedure is just the same as the original procedure since all agents must apply to L_v. Thus, a recursive execution is enough to solve the subproblem.

Execute the procedure to Example 2.

(Example 2 with ML) $n = 4$, $m = 4$, $c_1 = c_2 = c_3 = c_4 = 3$, $k_1 = k_2 = k_3 = k_4 = 3$, Every student's preference: $l_1 \succ l_2 \succ l_3 \succ l_4$. Every laboratory's preference: $s_1 \succ s_2 \succ s_3 \succ s_4$. $ML = [s_1, s_2, s_3, s_4]$.

(First round) Each agent applies according to the preference and each laboratory accepts according to the preference. Thus, l_1, l_2, and l_3 accept one agent of s_1, s_2 and s_3. Agents of s_4 are rejected from all of l_1, l_2, and l_3. l_4 accepts one agent of s_4. The remaining two agents of s_4 become vacant, since the remaining slots are in l_4. Therefore, $L_v = \{l_4\}$ and $va = 2$.

Thus, two agents are restricted to apply to L_v. From ML, one agent of s_1 and s_2 becomes the restricted agents. With the restrictions, execute the next round.

(Second round) All agents of s_3 and s_4, and two agents of s_1 and s_2 apply according to the preference. One agent of s_1 and s_2 must apply to l_4. Thus, l_1 and l_2 accepts one agent of s_1, s_2, and s_3. l_3 accepts one agent of s_3 and s_4. l_4 accepts one agent of s_1, s_2, and s_4. One of s_4's agents becomes a vacant agent. There is one vacant slot in l_3. Thus, the vacant laboratory set is increased, that is, $L' = \{l_3\}$ and $va = 1$.

After the second round, $L_v = \{l_3, l_4\}$ and $esum = 2 + 1 = 3$. Thus, three agents must be restricted to apply to L_v. Using ML, one agent is considered from s_1, s_2, and so on. s_1 already has one restricted agent, thus this agent is restricted to new L_v and e is increased. s_2 already has one restricted agent, thus this agent is restricted to new L_v and e is increased to 2. s_3 has one free agent accepted by L_v, thus even if this agent is restricted, vacant slots will not be decreased. Thus e is unchanged, but anyway, one agent in s_3 becomes a restricted agent. s_4 has a vacant agent, thus even if this agent is restricted, vacant slots will not be decreased. Thus e is unchanged, but anyway, one agent becomes a restricted agent. Every student in ML is used, thus next we consider s_1 again. One additional s_1's agent becomes a restricted agent. One agent of s_1 is accepted by $L - L_v$, thus this restriction is effective and e is increased. $e = esum = 3$, thus the restrictions are finished.

(Third round) Two agents of s_1, one agent of s_2, s_3, and s_4 must apply to $\{l_3, l_4\}$. All the other agents apply according to their preferences. Thus, l_1 accepts one agent of s_1, s_2 and s_3. l_2 accepts one agent of s_2, s_3 and s_4. l_3 accepts one agent of s_1, s_2 and s_3. l_4 accepts one agent of s_1 and s_4. One agent of s_4 becomes vacant.

There is no new vacant laboratory. In addition, the set of agents accepted by L_v and vacant agents are two of s_1, one of s_2, one of s_3, and two of s_4. This agent set is feasible for L_v. Thus, execute the original procedure with L_v and these agents.

(Fourth round(First round of subset assignment)) The assignment of $L - L_v$ is fixed and not changed anymore. Now, set new L as $\{l_3, l_4\}$ and execute the original procedure. The first round of this procedure

is just the same as in the previous round and l_3 accepts one agent of s_1, s_2, and s_3. l_4 accepts one agent of s_1 and s_4. One agent of s_4 becomes vacant.

Thus, in this execution, new $L_v = \{l_4\}$ and $va = 1$. A new restriction is necessary. Use ML from the beginning again. s_1 has a free agent accepted by L_v, thus the agent is restricted but e is unchanged. s_2 has a free agent accepted by $L - L_v$, thus this agent is restricted and set $e = 1$. The restriction is finished.

(Fifth round (Second round of subset assignment)) Execute again with this restriction. One agent of s_1 and s_2 must apply to l_4. l_3 accepts one agent of s_1, s_3, and s_4. l_4 accepts one agent of s_1, s_2, and s_4.

The final assignment is as follows: $\mu(l_1) = \{s_1, s_2, s_3\}$, $\mu(l_2) = \{s_2, s_3, s_4\}$, $\mu(l_3) = \{s_1, s_3, s_4\}$, $\mu(l_4) = \{s_1, s_2, s_4\}$.

(Example 3) $n = 4$, $m = 6$, $c_1 = c_2 = 2$, $c_3 = c_4 = 4$, $k_1 = k_2 = 2$, $k_3 = k_4 = 1$, $k_5 = k_6 = 3$. s_1, s_2, s_3, s_5, s_6's preference: $l_1 \succ l_2 \succ l_3 \succ l_4$. s_4's preference: $l_1 \succ l_2 \succ l_4 \succ l_3$. Every laboratory's preference: $s_1 \succ s_2 \succ s_3 \succ s_4 \succ s_5 \succ s_6$. $ML = [s_1, s_2, s_3, s_4, s_5, s_6]$.

(First round) All agents apply according to their preferences. l_1 accepts one agent of s_1 and s_2. Similarly, l_2 accepts one agent of s_1 and s_2. l_3 accepts one agent of s_3, s_5, and s_6. l_4 accepts one agent of s_4, s_5, and s_6. l_3 and l_4 have one vacant slot. s_5 and s_6 have one vacant agent. Thus, $L_v = \{l_3, l_4\}$ and $va = 2$.

Using ML, s_1 and s_2 have one agent restricted to L_v.

(Second round) One agent of s_1 and s_2 apply to l_1 and they are accepted. One agent of s_3 and s_4 apply to l_2 and they are accepted. One agent of s_1 and s_2 apply to l_3 and they are accepted. l_3 accepts one agent of s_5 and s_6. l_4 accepts one agent of s_5 and s_6. l_4 has two vacant slots. s_5 and s_6 have one vacant agent.

L_v is unchanged. Since the agents accepted by L_v and vacant agents form an infeasible problem of L_v assignment, va=2, thus $esum = 4$ and two more additional restricted agents are necessary. One agent of s_1, s_2, s_3, and s_4 must be restricted to L_v.

(Third round) One agent of s_1 and s_2 apply to l_1 and they are accepted. One agent of s_5 and s_6 are accepted by l_2. One agent of s_1, s_2, s_3, and s_5 are accepted by l_3 l_4 accepts one agent of s_4, s_5 and s_6. s_6 have one vacant agent.

Though there is a vacant slot, the agents accepted by L_v and the vacant agents form a feasible problem of L_v assignment. Thus no additional restrictions are necessary.

(Fourth round(First round of subset assignment)) The assignments of $L - L_v$ are not changed any more.

The assignment of new $L = \{l_3, l_4\}$ is executed using the same routine, but the first round is just the same as in the third round and l_3 accepts one agent of s_1, s_2, s_3 and s_5. l_4 accepts one agent of s_4, s_5 and s_6. s_6 have one vacant agent.

Thus, new $L_v = \{l_4\}$, $va = 1$, and one agents must be restricted. Using ML from the beginning and one agent of s_1 become restricted.

(Fifth round (Second round of subset assignment)) One agent of s_1 must apply to l_4. l_3 accepts one agent of s_2, s_3, s_5, and s_6. l_4 accepts one agent of s_1, s_4, s_5, and s_6.

The final assignment is as follows: $\mu(l_1) = \{s_1, s_2\}$, $\mu(l_2) = \{s_5, s_6\}$, $\mu(l_3) = \{s_2, s_3, s_5, s_6\}$, $\mu(l_4) = \{s_1, s_4, s_5, s_6\}$.

As shown above, we need to force some agents not to apply to their prefer laboratories, thus the matching result is not stable. For example, in the result of Example 2, s_1 and l_2 prefer (s_1, l_2) and (s_2, l_3) than current pairs (s_2, l_2) and (s_1, l_3). Since s_1's agent accepted by l_3 is a restricted agent, such cases are inavoidable to obtain a feasible result.

Instead, we define many-to-many justified stability using agents and their restriction statuses. Each agent has a type whether it is free or restricted. Since the algorithm is executed to $L_1(=L), L_2, \ldots, L_\alpha$ so that $L_{i+1} \subset L_i (1 \leq i \leq \alpha - 1)$ and the type is given in each execution, the type of each agent becomes a tuple $(t_1, t_2, \ldots, t_\alpha)$, where $t_i = free$ or $restricted$ in the last round of L_i's assignment. An agent accepted by a laboratory in $L_i - L_{i+1}$ is not involved in the assignment of L_{i+1}. For such cases, let $t_j (j > i) = \perp$. Let $t_i(s)$ be the i-th element of agent$s$'s type. In Example 2, the three agents of s_1 have types $(free, \perp)$, $(restricted, free)$, and $(restricted, restricted)$

A many-to-many matching μ is justified stable if there is no pair of agent and laboratory $(s_{i,j}, l)$ that satisfy the following conditions.

- $s_{i,j} \notin \mu(l)$.
- There is a pair $(s_{i',j'}, l')$ that satisfy the following properties: $s_i \succ_l s_j$, $l \succ_{s_i} l'$, $s_{i',j'} \in \mu(l)$, $l' = \mu(s_{i,j})$, and swapping the matching $(s_{i,j}, l')$, $(s_{i',j'}, l)$ to $(s_{i,j}, l)$, $(s_{i',j'}, l')$ results in a feasible matching.
- For some k, $t_k(s_{i,j}) = free$ and $t_k(s_{i',j'}) \neq \perp$ or $t_\alpha(s_{i,j}) = t_\alpha(s_{i',j'}) = restricted$.

The agents of a blocking pair and its supporting pair must be restricted to the above conditions. As for the previous example, the type of s_1's agent accepted by l_3 is $(restricted, free)$ and the type of s_2's agent accepted by l_2 is $(free, \perp)$ thus the pair of these agents is not the agent of a blocking pair (s_1's agent) and its supporting pair (s_2's agent).

Before showing that the matching is justified stable, we need to show some properties used for the proof. First, we show the properties related to enlargements of L_v.

LEMMA 3.1. *The number of vacant slots never increases during the iterations.*

(Proof) Introducing new restricted agents temporary makes unoccupied slots in $L - L_v$. The slots are occupied by (1)vacant agents or (2)free agents in L_v Case (1) decreases the number of vacant slots, though case (2) does not change the number of vacant slots. □

Note that the procedure terminates increasing the restricted agents when the feasibility test succeeds.

LEMMA 3.2. *The enlargement of L_v eventually terminates before L_v becomes L.*

(Proof) When an agent in $L - L_v$ is newly set as a restricted agent, there is a temporal vacant slot. The slot must be filled by some other agent. The possible cases are (1)vacant agent and (2)free agent in L_v. When (1) occurs, the number of vacant slots is decreased. When (2) occurs, the number of vacant slots is unchanged. When (1) occurs, there can be vacant slots outside of L_v and the enlargement of L_v might occur.

When a new student that had no vacant slots in the previous round has a vacant slot outside of L_v, the agent must be rejected by all laboratories, thus it must be rejected by all laboratories in L_v. Since the number of the restricted agents is set so that there is no overflow of agents in L_v occurs, it does not occur that an agent is newly rejected from all laboratories in L_v. Thus, when an enlargement of L_v occurs, the vacant agents are the ones in the first iteration ($z = 1$).

The size of L_v might increase during the iteration, but the size of L_v does not become to be equal to L. Since the student with a vacant agent has one agent accepted by each laboratory in L_v, $L_v = L$ means

that every laboratory has one accepted agent and there is at least one additional vacant agent. It is a contradiction that $k_i \leq n$ for any student. □

Next, we show that the feasibility test eventually succeeds.

LEMMA 3.3. *After the enlargement of L_v is finished, eventually the feasibility test succeeds in L_v.*

(Proof) After the enlargement of L_v is finished, if L_v is not feasible, new restrictions are set so that the number of accepted agents in L_v increases. If the number of restricted agents becomes the total number of slots, L_v becomes a feasible problem. Thus, the restriction adding also eventually terminates. □

Note that since the restriction is set to the students in a round-robin manner, the restriction does not result in an infeasible problem such that the number of restricted agents of a student becomes more than $|L_v|$.

For the types of agents, the following property holds.

LEMMA 3.4. *If $t_i(s) = free$, $t_{i+1}(s)$ is $free$ or $\perp$.*

(Sketch of proof) Since the number of vacant slots in L_{i+1} is less than that in L_i, the number of restricted agents in L_{i+1}'s assignment is less than that in L_i. □

THEOREM 3.5. *The matching given by the procedure has no justified blocking pair.*

(Proof) Let l' be the laboratory $s_{i,j}$ is accepted. Suppose that there is a pair $(s_{i,j}, l)$ and $(s_{i',j'}, l')$ that satisfies the condition of the justified blocking pair and its supporting pair.

(Case 1) For some k, $t_k(s_{i,j}) = free$ and $t_k(s_{i',j'}) \neq \perp$.

(Case 1-1) $t_k(s_{i,j})$ is accepted by $L_k - L_{k+1}$.

If $l \in L_k - L_{k+1}$, such a blocking pair does not exist since the last assignment of L_k is executed by the DA mechanism.

If $l \in L_{k+1}$, since there were vacant slots in L_{k+1} in the previous rounds, $s_{i,j}$ would have applied to l in a previous round and the application is accepted. Thus, such a pair does not exist.

(Case 1-2) $t_k(s_{i,j})$ is accepted by L_{k+1}.

Since $t_{k'}(s_{i,j})$ is $free$ or $\perp$ for $k' > k$, $s_{i,j}$ will be accepted by no worse laboratory for the agent in the further recursive assignments. The reason is as follows. In the first round of the assignment of L_{k+1}, (in which all agents become free) the result is the same and in the further rounds, some other agents become restricted, thus the result might become better for $s_{i,j}$.

If $l \in L_k - L_{k+1}$, such a blocking pair does not exist since the last assignment of L_k is executed by the DA mechanism and further assignments for $s_{i,j}$ are no worse than the one in the last assignment of L_k.

If $l \in L_{k+1}$, $t_k(s_{i',j'})$ is $free$ or *restricted*. In the assignments in L_{k+1}, $s_{i,j}$ remains $free$ and $s_{i',j'}$ is $free$ or *restricted*. Thus, by the argument of (Case 1-1), the result does not have a justified blocking pair.

(Case 2) $t_\alpha(s_{i,j}) = t_\alpha(s_{i',j'}) = restricted$.

There is no more recursion. In the last round of assignment of L_α, the assignment is done by the DA mechanism and both agents are restricted agents. Thus, the result does not have a justified blocking pair. □

The time complexity of the algorithm is polynomial of $\sum_i k_i$, the total number of slots. The number of rounds is less than n and the number of recursive executions is less than n. The time complexity of each execution of DA is at most $O(n * \sum_i k_i)$. Thus the complexity is at most $O(n^3 * \sum_i k_i)$.

4 CONCLUSION

This paper discussed the many-to-many perfect matching problem, which can be used as a matching between multiple major students and laboratories. We showed the DA mechanism that uses ML as a tie-breaking between students. One further study is a relaxed problem in which some number of vacant slots in laboratories are allowed, but students must be assigned to a fixed number of different laboratories.

REFERENCES

[1] Mourad Baıou and Michel Balinski. 2000. Many-to-many matching: stable polyandrous polygamy (or polygamous polyandry). *Discrete Applied Mathematics* 101, 1-3 (2000), 1–12.

[2] Katarína Cechlárová, Pavlos Eirinakis, Tamás Fleiner, Dimitrios Magos, David F Manlove, Ioannis Mourtos, Eva Ocel'áková, and Baharak Rastegari. 2015. Pareto optimal matchings in many-to-many markets with ties. In *International Symposium on Algorithmic Game Theory*. Springer, 27–39.

[3] Ning Chen and Mengling Li. 2019. Pareto stability in two-sided many-to-many matching with weak preferences. *Journal of Mathematical Economics* 82 (2019), 272–284. https://doi.org/10.1016/j.jmateco.2019.03.005

[4] Federico Echenique and Jorge Oviedo. 2006. A theory of stability in many-to-many matching markets. *Theoretical Economics* 1 (2006), 233–273.

[5] Pavlos Eirinakis, Dimitrios Magos, Ioannis Mourtos, and Panayiotis Miliotis. 2012. Finding all stable pairs and solutions to the many-to-many stable matching problem. *INFORMS Journal on Computing* 24, 2 (2012), 245–259.

[6] Daniel Fragiadakis, Atsushi Iwasaki, Peter Troyan, Suguru Ueda, and Makoto Yokoo. 2016. Strategyproof matching with minimum quotas. *ACM Transactions on Economics and Computation (TEAC)* 4, 1 (2016), 1–40.

[7] David Gale and Lloyd S Shapley. 1962. College admissions and the stability of marriage. *The American Mathematical Monthly* 69, 1 (1962), 9–15.

[8] Masahiro Goto, Atsushi Iwasaki, Yujiro Kawasaki, Ryoji Kurata, Yosuke Yasuda, and Makoto Yokoo. 2016. Strategyproof matching with regional minimum and maximum quotas. *Artificial intelligence* 235 (2016), 40–57.

[9] John William Hatfield, Fuhito Kojima, and Yusuke Narita. 2014. Many-to-many matching with max-min preferences. *Discrete Applied Mathematics* 179 (2014), 235–240. https://doi.org/10.1016/j.dam.2014.07.003

[10] Robert W Irving and Sandy Scott. 2007. The stable fixtures problem– A many-to-many extension of stable roommates. *Discrete Applied Mathematics* 155, 16 (2007), 2118–2129.

[11] Paula Jaramillo, Ça 認 atay Kayı, and Flip Klijn. 2014. On the exhaustiveness of truncation and dropping strategies in many-to-many matching markets. *Social Choice and Welfare* 42, 4 (2014), 793–811.

[12] Zhenhua Jiao and Guoqiang Tian. 2015. The stability of many-to-many matching with max-min preferences. *Economics Letters* 129 (2015), 52–56.

[13] Zhenhua Jiao and Guoqiang Tian. 2017. The blocking lemma and strategy-proofness in many-to-many matchings. *Games and Economic Behavior* 102 (2017), 44–55.

[14] Naoyuki Kamiyama. 2019. Many-to-many stable matchings with ties, master preference lists, and matroid constraints. In *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems*. 583–591.

[15] Bettina Klaus and Flip Klijn. 2017. Non-revelation mechanisms for many-to-many matching: Equilibria versus stability. *Games and Economic Behavior* 104 (2017), 222–229.

[16] Flip Klijn and Ayşe Yazıcı. 2014. A many-to-many 'rural hospital theorem'. *Journal of Mathematical Economics* 54 (2014), 63–73.

[17] Hideo Konishi and M Utku Ünver. 2006. Credible group stability in many-to-many matching problems. *Journal of Economic Theory* 129, 1 (2006), 57–80.

[18] László Lovász and Michael D Plummer. 2009. *Matching theory*. Vol. 367. American Mathematical Soc.

[19] Ruth Martınez, Jordi Massó, Alejandro Neme, and Jorge Oviedo. 2004. An algorithm to compute the full set of many-to-many stable matchings. *Mathematical Social Sciences* 47, 2 (2004), 187–210.

[20] Yasunori Okumura. 2017. A one-sided many-to-many matching problem. *Journal of Mathematical Economics* 72 (2017), 104–111.

[21] Antonio Romero-Medina and Matteo Triossi. 2021. Two-sided strategy-proofness in many-to-many matching markets. *International Journal of Game Theory* 50, 1 (2021), 105–118.

[22] Antonio Romero-Medina and Matteo Triossi. 2022. Take-it-or-leave-it contracts in many-to-many matching markets. *Economic Theory* (2022), 1–33.

[23] Jay Sethuraman, Chung-Piaw Teo, and Liwen Qian. 2006. Many-to-one stable matching: geometry and fairness. *Mathematics of Operations Research* 31, 3 (2006), 581–596.