SHANDONG UNIVERSITY

School of Computer Science and Technology

Institute of Software and Theoretical Computer Science

Matching Problem of Preference Model

Author :
Jiandong Liu

Supervisor :
Prof. Jiong Guo
Ph.D Yinghui Wen

January 10, 2020

Contents

1 Abstract

We introduce a matching problem of preference model based on four evaluation criteria which are derived from the generalized version of the famous STABLE MARRIAGE problem with multi-modal preference lists. The central twist herein is to allow each agent to rank its potentially matching counterparts based on more than one evaluation mode. Therefore, each agent is equipped with multiple preference lists, each ranking the counterparts in a possibly different way. We removed the previous stable matching constraints, proposed four new evaluation criteria to measure the stability of the matching and focus on computational complexity aspects in these new scenarios.

2 Introduction

In today's environment, there are more and more evaluation factors, so it is more meaningful to turn the original single-layer matching problem (affected by only one factor) into multiple layers for consideration. In the classic (conservative) STABLE MARRIAGE problem, we are given two disjoint sets U and W of n agents each, where each of the agents has a strict preference list that ranks every member of the other set. In addition, the goal is to find a bijection (what we call a matching) between U and W without any blocking pair which may endanger the stability of the matching. A blocking pair means that they are not matched to each other while rank each other higher than their respective partners in the matching.

Take the Figure 1 for example, there are four agents from two disjoint sets U and W with their strict preference. Based on the judgment criteria of the blocking pair, we could find that the $Matching_1$ is stable while the second is not.

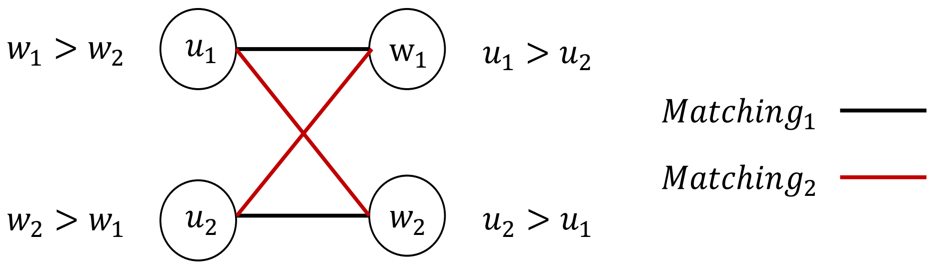


Figure 1: Two Matchings of Single-Layer

Moreover, we continually add a new factor to extend this single-layer model into a multi-layer model shown below. In the Figure 2, we could easily find that the $Matching_1$ is stable in $Layer_1$ while unstable in $Layer_2$ and the $Matching_2$ is exactly the opposite. Therefore, there is no global stable matching in this multi-layer model which is stable in each layer.

What's more, if we continue to study the multi-layer stable matching model, we could acquire the fact that the match is fixed for some prominent points. In other words, if there exists a most popular point which ranks first in each agent's preference list in a certain layer, the counterpart of this prominent point is fixed. Because if the one ranks first in the preference list of the prominent point does not match with it, the matching in this layer is unstable.

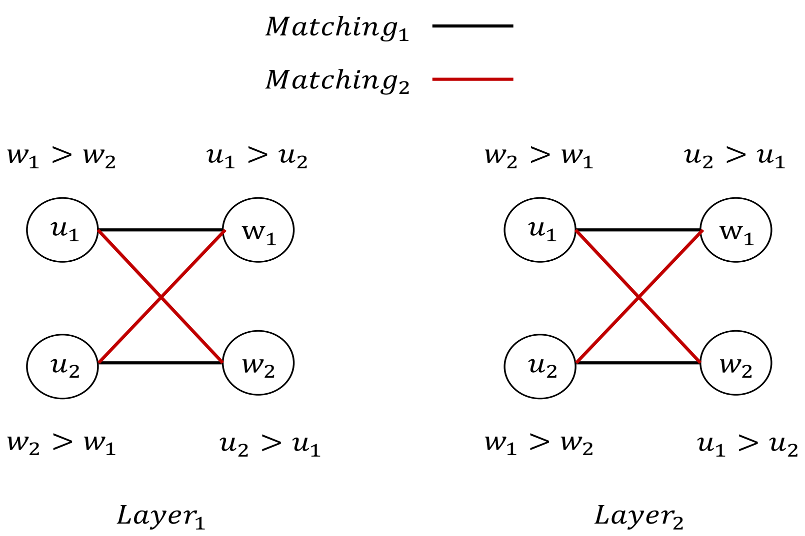


Figure 2: Two Matchings of Multi-Layer

Due to the two features of the multi-layer stable matching described above, we decide to remove the stable matching constraints and propose four new evaluation criteria to measure the stability of the matching and focus on computational complexity aspects.

3 Definitions

3.1 Basic Definitions

For each natural number t by $[t]$, we denote the set $\{1, 2, ..., t\}$.

Let $U = \{u_1, ..., u_n\}$ and $W = \{w_1, ..., w_n\}$ be two disjoint sets which contain n elements. There are l (a non-negative integer) layers of preferences. For each $i \in [l]$ and each $u \in U$, let $\succ_u^{(i)}$ be a linear order on W that represents the ranking of agent u over all agents from W in layer i. Similarly, for each $i \in [l]$ and each $w \in W$, the symbol $\succ_w^{(i)}$ represents a linear order on U that encodes preferences of w in layer i. We utilize the concept, preference lists, to represent such linear orders. A preference profile P_i of layer $i \in [l]$ is a set of preference lists of each agent in layer i, $\{\succ_a^{(i)} | a \in U \cap W\}$.

Let $U \star W = \{\{u, w\} | u \in U \land w \in W\}$. A matching $M \subseteq U \star W$ is a set of pairwisely disjoint pairs, *i.e.* for each two pairs $p, p' \in M$ it holds that $p \cap p' = \emptyset$.

If $\{u, w\} \in M$, then we also use $M(u)$ to represent w and $M(w)$ to represent u, and we hold that u and w are their respective partners under M; otherwise we say that $\{u, w\}$ is an unmatched pair.

3.2 Four Evaluation Criteria

In this part, we will introduce four evaluation criteria to judge the stability of a matching. We define the satisfaction of an agent x with respect to a given matching as the *rank* of its partner y assigned by this matching. Therefore, considering the minimum sum of the ranks of all partners, we propose the first criterion, *egalitarian cost*.

$$\textit{egal-cost}(M) := \sum_{\{u,w\} \in M} (rank_u(w) + rank_w(u))$$

What's more, in terms of the minimum maximum rank of any partner, we bring forward the *regret cost*.

$$\textit{regret-cost}(M) := \max_{i \in V(M)} rank_i(M(i))$$

Moreover, considering the minimum sum of absolute value of the difference between the ranks of one side and the minimum maximum of the sums of the ranks of one side, we propose the *sex-equal cost* and the *balance cost*.

$$\begin{aligned} \textit{equal-cost}(M) &:= \sum_{(u,w) \in M} |rank_u(w) - rank_w(u)| \\ \textit{balance-cost}(M) &:= max\{\sum_{(u,w) \in M} rank_u(w), \sum_{(u,w) \in M} rank_w(u)\} \end{aligned}$$

3.3 Global Layer and α-Layer Minimum Cost

According to the four evaluation criteria, we need to find a matching M whose *cost* is less than D, *i.e.* $cost(M) \leq D$ holds. Furthermore, for multi-layer model, there are two types of Minimum Cost, global layer and α-layer minimum cost.

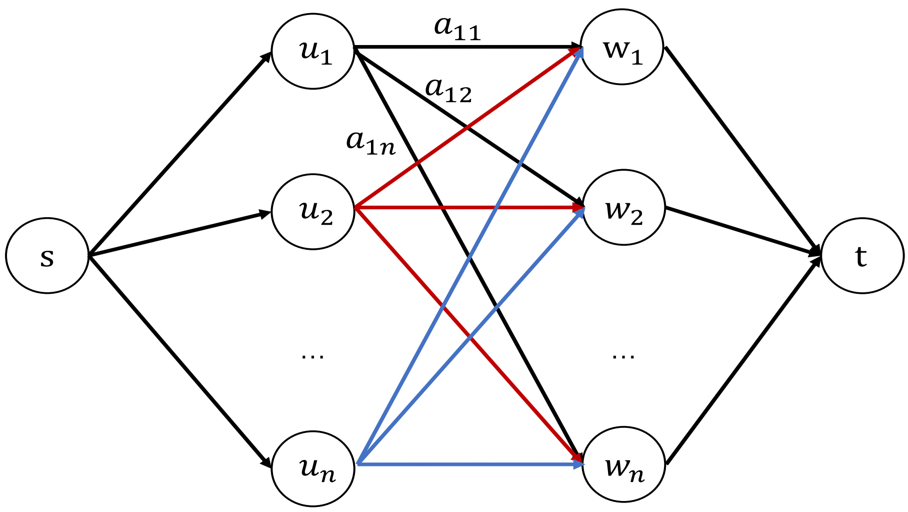
For the global layer cost, the goal is to find a matching M whose sum of cost in each layer is less than D. In addition, in terms of the α-layer cost, the goal is to find a matching M whose sum of cost in certain α layers chosen by you is less than D.

4 Egalitarian Cost

$$egal\text{-}cost(M) := \sum_{\{u,w\}\in M} (rank_u(w) + rank_w(u)) \leq D$$

4.1 Global Layer

In order to find a matching which minimize the Egalitarian Cost, we could change the original problem into bipartite graph. In addition, we could assign values to each edge which equals to $rank_u(w) + rank_w(u)$. Thus, the original problem of finding the minimum Egalitarian Cost is changed into the famous classic bipartite graph weighted matching problem which could be solved by the KM algorithm or cost flow.



$$a_{ij} = \sum_{k=1}^{k\leq l} (rank_{u_i}^k(w_j) + rank_{w_j}^k(u_i))$$

Figure 3: Cost Flow of Global Layer

In the bipartite graph Figure 3, each edge holds two values, flow and cost. Each edge starts from s or ends at t holds $flow = 1$ and $cost = 0$, while other edge connecting U and W holds $flow = 1$ and $cost = a_{ij}$. In addition, $rank_{u_i}^k(w_j)$ means the order of w_j in the preference list of u_i in the k^{th} layer.

Moreover, we choose the cost flow algorithm to calculate the exact matching satisfying the requirements. In the cost flow algorithm, every time we find a minimum cost path from s to t using the famous Dijkstra or Bellman-Ford algorithm. In addition, because the maximum flow is n and every time we find a minimum cost path, the flow will add one, the time complexity is $O(n^3 logn)$.

Besides, we still need to ensure that no negative loops occur during algorithm operation. Suppose a negative loop occurs in the algorithm like Figure 4, $a_{nn} + a_{21} < a_{2n} + a_{11}$ which means $\max(a_{2n}, a_{11}) > \min(a_{nn}, a_{21})$. Assume $a_{nn} = \min(a_{nn}, a_{21})$ and $a_{2n} = \max(a_{2n}, a_{11})$. However, according to the cost flow algorithm, every time we choose a minimum cost path. The path a_{2n} is chosen before a_{nn}, so $a_{2n} \le a_{nn}$ which contradicts $a_{2n} > a_{nn}$. Therefore, there is no negative loop occur during cost flow algorithm.

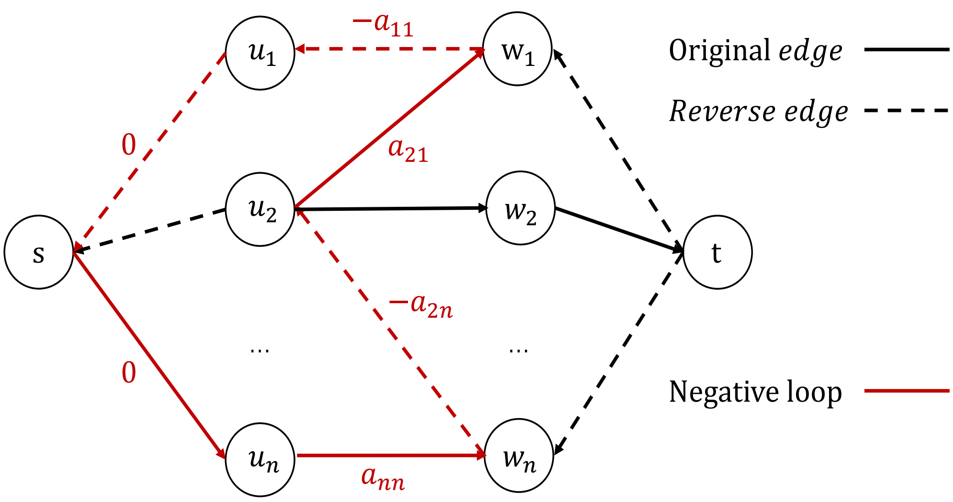


Figure 4: Negative Loop of Global Layer

4.2 α-Layer

In terms of the α-layer cost, we could prove it is NP-hard by reducing the 1-IN-3SAT problem which is a famous NPC problem to this problem. Firstly, we need to introduce the 1-IN-3SAT problem which is the following.

INSTANCE : A collection of clauses $C_1, ..., C_m, m > 1$; each C_i is a disjunction of exactly three literals.

QUESTION : Is there a truth assignment to the variables occuring so that exactly one literal is true in each C_i?

Example: Let $X = \{x_1, ..., x_5\}$ be the set of variables. Let the clause set $C = \{C_1, C_2, C_3\}$ be the following : $C_1 = \{\overline{x_1}, \overline{x_2}, x_3\}, C_2 = \{\overline{x_1}, x_4, x_5\}, C_3 = \{\overline{x_2}, x_4, x_5\}$. With this (X, C) we consider the 1-IN-3SAT problem. We say a clause is correctly satisfied if and only if the clause is satisfied due to exactly one literal in it.

After comprehending the famous NPC problem, 1-IN-3SAT, we could construct the original input data which could be reduced by the 1-IN-3SAT problem. According to the definition of the 1-IN-3SAT problem, there is n variables and m clauses which contain three literals and only one of the three literals is true. Thus, let $l = 3m$ and $\alpha = m$ and the goal is to choose α layers in l layers to acquire the egalitarian cost, $D = 6 * m * (n - 1)$. What's more, there is $2n$ agents each side while one variable represents four agents.

Assume that there is a clause $C_1 = \{x_1, x_2, x_3\}$ and we will construct the three layers. The Figure 5 is just one of three layers $(x_1 = 1, x_2 = 0, x_3 = 0)$ for one clause $C_1 = \{x_1, x_2, x_3\}$.

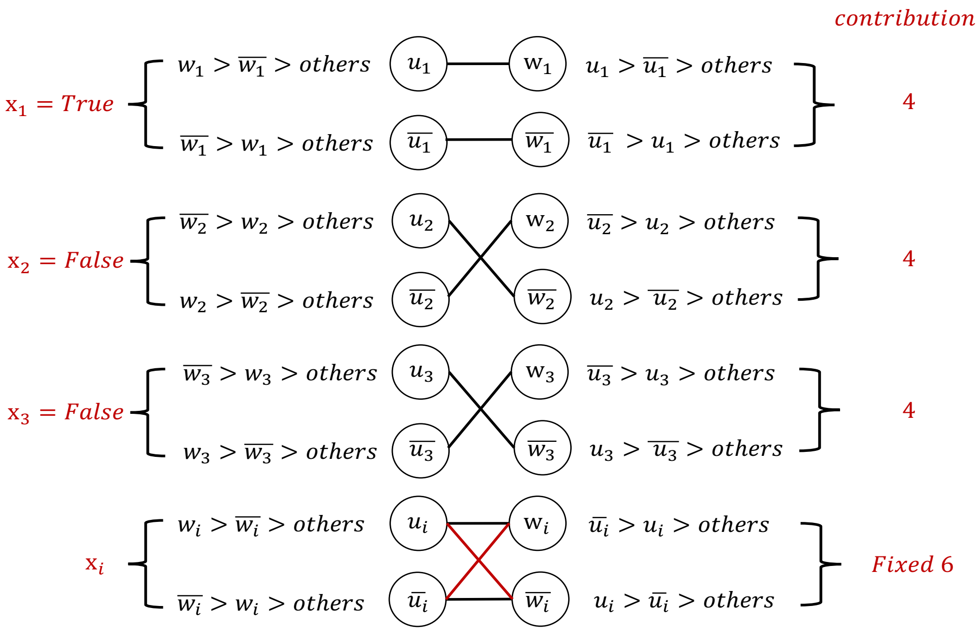


Figure 5: Constructing for One Clause

There are two more layers here, $(x_1 = 0, x_2 = 1, x_3 = 0)$ and $(x_1 = 0, x_2 = 0, x_3 = 1)$. Therefore, we could just choose one of the three layers to acquire the smallest egalitarian cost which equals to D. Consequently, we could reduce the 1-IN-3SAT problem to the constructing problem described above which proves the original problem is NP-hard.

5 Regret Cost

$$regret\text{-}cost(M) := \max_{i \in V(M)} rank_i(M(i)) \leq D$$

5.1 Global Layer

Similarly, we change the original problem into bipartite graph. The first one is focusing on the sum of cost while this is the maximal cost. Thus, we could use the Maximum flow algorithm to find a matching satisfying the requirements.

While constructing the bipartite graph, we assign $\max(rank_u(w), rank_w(u))$ to be the value of the edge. If the value is more than D, remove the edge.

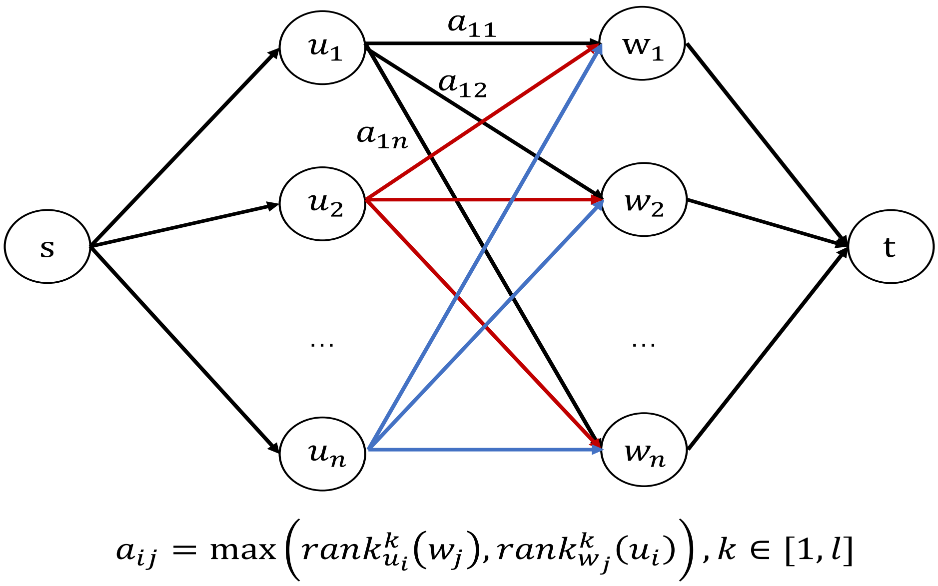


Figure 6: Maximum Flow of Global Layer

Different from the cost flow, the edge in maximum flow only holds one value which represents the flow. What's more, in the initial data, the flow of the edge starts from s or ends at t is one.

In this algorithm, every time we find a path using bfs or dfs algorithm. In addition, after each augmentation, the total flow adds 1 which means the number of augmentation is n. Therefore, the time complexity is $O(n^3)$.

5.2 α-Layer

In terms of the regret-cost, we could firstly remove the edge whose value is larger than D for each layer. Then check each layer if we could use the remaining edges to achieve an exact match. If not, remove the layer.

After the initial operation described above, the task left to us is find α layers whose intersection of edges could achieve an exact match. At first, we try to reduce the vertex coverage problem to this problem through construction, but we failed. Because vertex coverage is based on union, while this problem is based on intersection.

Furthermore, we tried a lot of common npc problems and attempt to reduce them to this problem, but all failed. Therefore, we still have no clue on this issue, but we will continue to find ways to accomplish this.

6 Equal Cost

$$\textit{equal-cost}(M) := \sum_{(u,w)\in M} |rank_u(w) - rank_w(u)| \leq D$$

6.1 Global Layer

Actually, this problem is not much different from the first one, so we only need to modify the model of the first problem to complete the solution of this problem.

Similarly, we transform the original problem into bipartite graph. The first one is focusing on the sum of addition while this is focusing on subtraction. Thus, we could use the cost flow algorithm to find a matching satisfying the requirements.

According to the Figure 7, we could easily connect it with the first one and consider if there exists negative loop during the operating process. However, the value of edge in this model is positive which means it is the same as the first one. Due to the proving process in the first part, we could ensure there is not negative loop during the operating process. What's more, the time complexity is $O(n^3 logn)$.

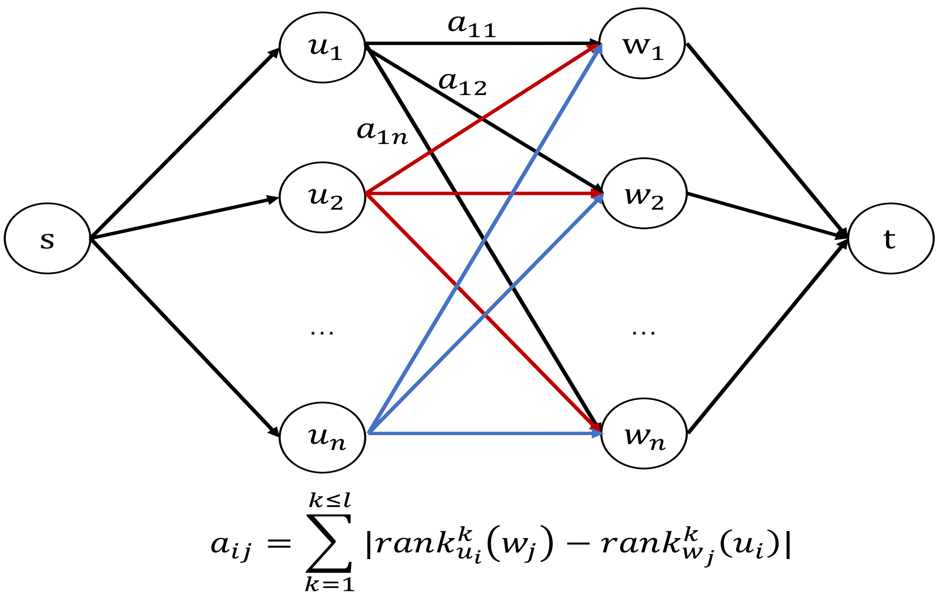


Figure 7: Equal Cost of Global Layer

6.2 α-Layer

Unlike the global layer, in this equal-cost problem of α-layer, there is slightly different from the first problem, and requires some modifications on the constructed model.

According to the understanding of the 1-IN-3SAT problem, we could construct the original input data which could be reduced by this NPC problem. In terms of the definition of the 1-IN-3SAT problem, there is n variables and m clauses which contain three literals and only one of the three literals is true. Thus, let $l = 3m$ and $\alpha = m$ and the goal is to choose α layers in l layers to acquire the egalitarian cost, $D = 0$. What's more, there is $2n$ agents each side while one variable represents four agents.

Assume that there is a clause $C_1 = \{x_1, x_2, x_3\}$ and we will construct the three layers. The Figure 8 is just one of three layers $(x_1 = 1, x_2 = 0, x_3 = 0)$ for one clause $C_1 = \{x_1, x_2, x_3\}$.

According to the constructing method, we could find that there are two more layers here, $(x_1 = 0, x_2 = 1, x_3 = 0)$ and $(x_1 = 0, x_2 = 0, x_3 = 1)$. Consequently, we could just choose one of the three layers to acquire the smallest equal cost which equals to D. Therefore, we could reduce the 1-IN-3SAT problem to the constructing problem described above which proves the equal cost problem with α layer is NP-hard.

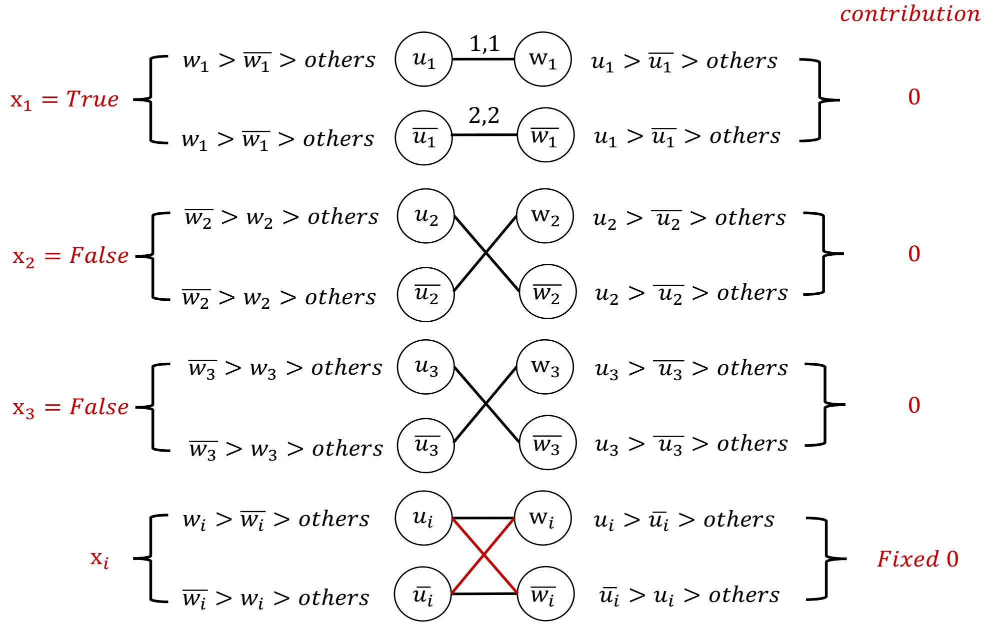


Figure 8: Equal Cost of α-Layer

7 Balance Cost

$$balance\text{-}cost(M) := max\{\sum_{(u,w)\in M} rank_u(w), \sum_{(u,w)\in M} rank_w(u)\} \leq D$$

7.1 Global Layer

Unlike other cost model, balance cost model cannot convert the two weights on the edge into one value, so it is more difficult than the previous models. At first, we still wanted to solve this problem on the bipartite graph, and designed an augmentation algorithm for the problem with two variables on the edge. But after programming and testing, it was found that the designed two-variable augmentation algorithm would have a negative loop, so the augmentation algorithm could not be executed correctly.

As the research continues, we find that the balance cost problem of global layer on the bipartite graph under the broad definition (that is, the order of edge weights is not strictly required) is an NP-hard problem, which can be reduced by the classic NPC problem reduction protocol.

It should be noted here that we only prove that the bipartite graph under the generalized condition is NP-hard, but we could only reduce the original problem to the general bipartite graph problem, and we cannot reduce the bipartite graph problem under the generalized condition to the original problem. Therefore, we cannot prove that the original problem is also NP-hard through the NP-hard bipartite graph under generalized conditions.

Consequently, we still haven't made enough progress to address this problem under the global layer. But we are still trying to return to the original model and give up the idea of bipartite graph conversion for further research.

7.2 α-Layer

According to the proving process of first criterion, we could just change the contribution in Figure 5 to obtain the model of this problem.

Similarly, we set the initial data including m, l, α which is the same as other cost model. Then we modify the contribution in this balance cost model and acquire the balance cost, $D = 3 * m * (n - 1)$.

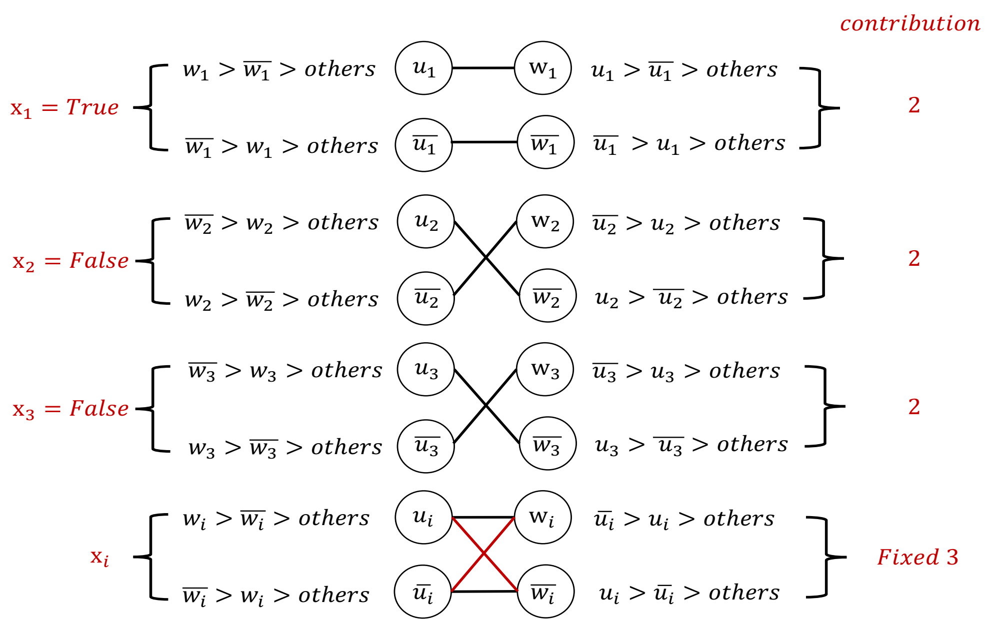


Figure 9: Balance Cost of α-Layer

Assume that there is a clause $C_1 = \{x_1, x_2, x_3\}$ and we will construct the three layers. The Figure 9 is just one of three layers $(x_1 = 1, x_2 = 0, x_3 = 0)$ for one clause $C_1 = \{x_1, x_2, x_3\}$.

There are two more layers here, $(x_1 = 0, x_2 = 1, x_3 = 0)$ and $(x_1 = 0, x_2 = 0, x_3 = 1)$. Therefore, we could just choose one of the three layers to acquire the smallest balance cost which equals to D. Consequently, we could reduce the 1-IN-3SAT problem to the constructing problem described above which proves the original problem is NP-hard.

8 Conclusion

According to the four models described above, we could summarize our contributions to this Matching Problem of Preference Model.

Table 1: Complexity Analysis of the Four Matching Models

criterion	Global Layer	α-Layer
Egalitarian Cost	$O(n^3 logn)$	NP-hard
Regret Cost	$O(n^3)$	NP-hard
Equal Cost	$O(n^3 logn)$	Studying
Balance Cost	Studying	NP-hard

References

[1] Jiehua Chen.Computational Complexity of Stable Marriage and Stable Roommates and Their Variants[R].Beer-Sheva, Israel:University of Warsaw,2019.

[2] Jiehua Chen,Rolf Niedermeier,Piotr Skowron.Stable Marriage with Multi-Modal Preferences[R].Berlin, Germany:TU Berlin,2018.