k-Best Egalitarian Stable Marriages for Task Assignment

Siyuan Wu
University of Macau
Macau SAR, China
asuka.wsy@connect.umac.mo

Leong Hou U
University of Macau
Macau SAR, China
ryanlhu@um.edu.mo

Panagiotis Karras
Aarhus University
Denmark
piekarras@gmail.com

ABSTRACT

In a two-sided market with each agent ranking individuals on the other side according to their preferences, such as location or incentive, the *stable marriage* problem calls to find a *perfect matching* among the two sides such that no pair of agents prefers each other to their assigned matches. Recent studies show that the number of solutions can be large in practice. Yet the classic solution by the Gale-Shapley (GS) algorithm is *optimal* for agents on the one side and *pessimal* for those on the other side. Some algorithms find a stable marriage that optimizes a measure of the cumulative satisfaction of all agents, such as *egalitarian cost*. However, in many real-world circumstances, a decision-maker needs to examine a set of solutions that are stable and attentive to both sides and choose among them based on expert knowledge. With such a disposition, it is necessary to identify a set of high-quality stable marriages and provide transparent explanations for any reassigned matches to the decision-maker. In this paper, we provide efficient algorithms that find the k-best stable marriages by *egalitarian cost*. Our exhaustive experimental study using real-world data and realistic preferences demonstrates the efficacy and efficiency of our solution.

PVLDB Reference Format:
Siyuan Wu, Leong Hou U, and Panagiotis Karras. k-Best Egalitarian Stable Marriages for Task Assignment. PVLDB, 16(11): 3240 - 3252, 2023.
doi:10.14778/3611479.3611522

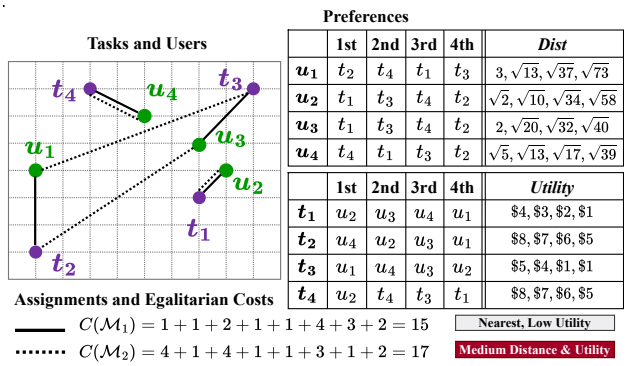
PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at github.com/Asuka54089/KESMP.

1 INTRODUCTION

The problem of *task assignment* finds application in crowdsourcing, resource allocation, and decision support [5, 32, 47, 52, 53, 59, 61]. Previous work usually considers the problem under a *symmetric* function of cost or preference, on which both sides agree [31, 48, 61]. However, in spatial assignment applications the problem often arises with an *asymmetric* cost function, by which agents on the two sides evaluate each other by divergent criteria, e.g., distance from vehicles or travel cost on the one side vs. interest in tasks or utility gained on the other side [3, 12]. The same predicament also appears in other task assignment applications. For example, university admissions rely on students' preferences for college places and their assessment by colleges [12], while in online recruitment employers and candidates operate by divergent criteria [56].

The Stable Marriage Problem (SMP) [12] is defined on a market comprising two disjoint sides dubbed *men* and *women*. Every woman (man) ranks the men (women) on the other side by preference. A solution matches agents on the two sides to each other; a matching is called *stable* if there is no pair of a man and a woman who would rather be matched to each other than to their assigned matches. SMP has been extensively studied for decades with applications including the allocation of residents to hospitals [1, 19], students to colleges [2, 12, 43], and tasks to workers [31, 61].

However, the consideration of two-sided preferences leads to a problem of *fairness* among the two sides. As the preferences on the two sides are often unrelated, it is challenging to find a solution that satisfies both sides. The Gale-Shapley (GS) algorithm [39] computes a stable marriage in $O(n^2)$ time, yet assigns to each agent on one side the *best*, and to each one on the other side the *worst* match they could get in any stable solution. Several works have studied the ensuing *fairness* question [9, 19, 24, 27, 38] aiming to optimize an equity cost, as there does not exist a dominant solution that simultaneously achieves quality and fairness. Among them, the *Egalitarian* Stable Marriage Problem (ESMP) [18] calls to minimize the cumulative dissatisfaction of agents, or *egalitarian cost*.



Preferences

	1st	2nd	3rd	4th	*Dist*
u_1	t_2	t_4	t_1	t_3	$3, \sqrt{13}, \sqrt{37}, \sqrt{73}$
u_2	t_1	t_3	t_4	t_2	$\sqrt{2}, \sqrt{10}, \sqrt{34}, \sqrt{58}$
u_3	t_1	t_3	t_4	t_2	$2, \sqrt{20}, \sqrt{32}, \sqrt{40}$
u_4	t_4	t_1	t_3	t_2	$\sqrt{5}, \sqrt{13}, \sqrt{17}, \sqrt{39}$

	1st	2nd	3rd	4th	*Utility*
t_1	u_2	u_3	u_4	u_1	\$4, \$3, \$2, \$1
t_2	u_4	u_2	u_3	u_1	\$8, \$7, \$6, \$5
t_3	u_1	u_4	u_3	u_2	\$5, \$4, \$1, \$1
t_4	u_2	t_4	t_3	t_1	\$8, \$7, \$6, \$5

Figure 1: An example of the 2-best egalitarian stable marriages in a task-to-user market.

However, merely minimizing egalitarian cost may still result to a solution that is excessively favorable to one side and exceedingly harsh to the other side. We use a compelling example of spatial task assignment to illustrate our problem in Figure 1.

Example 1 (The k-best egalitarian stable marriages in spatial task assignment). *Figure 1 shows a two-sided market of tasks and users, where users prefer nearby tasks and tasks require users of high utility on the task. For example, user u_1 prefers task t_2 to t_4, as the distance from u_1 to t_2 is 3 but to t_4 is $\sqrt{13}$. Similarly, users u_2*

Proceedings of the VLDB Endowment, Vol. 16, No. 11 ISSN 2150-8097.
doi:10.14778/3611479.3611522

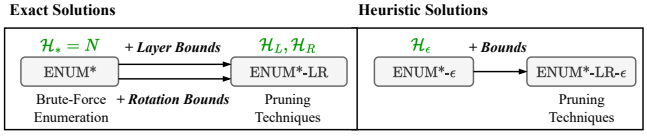
and u_3 are the best-2 choices for task t_1, since u_2 brings a benefit of \$4 while u_3 produces \$3. The tables in the figure list the full preference lists. This instance has two stable marriages, $\mathcal{M}_1$ and $\mathcal{M}_2$, with egalitarian costs 15 and 17 (i.e., the sum of all matches' positions), respectively. Even though $\mathcal{M}_1$ has lower egalitarian cost, it assigns t_2 to u_1, which is optimal in terms of distance but pessimal in terms of utility. $\mathcal{M}_2$ may be a better choice if t_2 does not accept u_1 due to the low utility. Besides, $\mathcal{M}_1$ is more favorable to users, as $C(\mathcal{M}_1) = 5+10$, while $\mathcal{M}_2$ is more favorable to tasks, as $C(\mathcal{M}_2) = 10+7$. Thus, it is tough to make a choice merely by egalitarian cost.

Since these quality and fairness issues are hard to be formulated precisely and objectively, presenting only a single best solution is insufficiently dependable. From a technological perspective, it is difficult to simulate more complex situations involving political and social considerations, which are considered by decision-makers when making a final plan. For example, in the paper-reviewer assignment process, the program chair may need to adjust the assignment results generated by conference management tool [35, 36]. A decision maker may thus require a set of high-quality stable marriages [8] to determine one that best meets their requirements.

Let the set of all stable marriages in a problem instance be $\mathbb{M}$. Given the multiplicity of solutions, a stakeholder may wish to examine the k best stable marriages, $\mathbb{M}(k) = \{\mathcal{M}^1, \mathcal{M}^2, \ldots, \mathcal{M}^k\}$, where $\mathcal{M}^i \in \mathbb{M}(k) \subseteq \mathbb{M}$ is at position i by egalitarian cost $C(\mathcal{M}^i)$, and choose one based on domain expertise and other desiderata. This kind of problem, looking for the k best solutions by some criterion, has been studied in assignment tasks, e.g., k best matchings in weighted graphs [4, 40], and in discrete optimization problems, e.g., k shortest paths [7, 57] and k-nearest-neighbor search in spatial database [41, 51]. However, to our knowledge, the problem of finding the k-best egalitarian stable marriages has yet to be considered.

In this paper, we study the problem of finding the k-best stable marriages by egalitarian cost, or *the k-best Egalitarian Stable Marriages Problem* (k-ESMP). To our knowledge, this is the first work to study k-ESMP. Previous studies have aimed to find optimal stable marriages under constraints [15, 26] and fair matchings [14, 17, 46, 49, 50, 54, 55], but have not considered k-ESMP, which, as we show, is **NP**-hard. Our solution returns the k-best stable marriages as opposed to a single-assignment solution, thus enables a human-in-the-loop approach that lets decision-makers to participate in the process. We offer alternatives to the decision maker in the form of re-assigned matches, which are efficiently represented by a compact structures known as *rotations* [19, 21, 23]. We present a case study in our experimental section.

A brute-force solution to k-ESMP is to *enumerate* all stable marriages and return the top ones by egalitarian cost. An iterative algorithm [18] enumerates all stable marriages in $O(n^2 + nN)$ time, where n is the number of agents in one side and N is the number of all possible stable marriages. However, this approach is impracticable, as the number of stable marriages in real-world markets may be extremely large in practice, since the number of agents in two-sided markets increases in real applications while some real-world stable marriage instances inherently generate exceptionally many stable solutions [3, 20] and up to exponential in the number of agents in theory [23]. To reduce this search space, we devise and use two bounds, the *layer bound* and the *rotation bound*, to eschew unpromising enumeration branches; we also propose a heuristic for k-ESMP. Figure 2 outlines our contributions.



(a) Solutions

Solution	Search Space
ENUM*-L	$\mathcal{H}_L = N - \sum_{\alpha=1}^{n/2} \binom{\lvert V^\ell \rvert}{\alpha}$
ENUM*-R	$\mathcal{H}_R = N - \sum_{\alpha=1}^{n/2} \binom{\lvert R \rvert}{\alpha}$
ENUM*-ϵ	$\mathcal{H}_\epsilon = \sum_{\alpha=1}^{n/2} \binom{\lvert R^{\geq *} \rvert}{\alpha}$

(b) Search Spaces

Figure 2: Our Solutions and their search spaces.

The rest of this paper is structured as follows. We introduce related work in Section 2 and preliminaries in Section 3. Section 4 defines the k-ESMP and discusses our solutions. In Section 5 we present our experimental results and we conclude in Section 6.

2 RELATED WORK

Spatial Task Assignment. Spatial task assignment is a cardinal problem in spatial crowdsourcing [32]. It aims to find a task-to-worker assignment among spatial data subject to constraints. While most previous works focus on *symmetric* preferences among workers and tasks [31, 61], a few studies have examined the more realistic case of *asymmetric* preferences of workers and tasks for the opposite side, based on, e.g., the distance or reputation of a task [61], the waiting time for a customer, or idle time of a driver [31], and aim to optimize a global objective under constraints considering the satisfaction of both workers and tasks. The work in [47] defines the preferences of workers by an order-based model and finds a matching that minimizes a total distance cost. The CA-SC problem [6] defines the cooperation quality score among workers as the optimization goal. The FETA problem [5] aims to balance the waiting time among workers based on the concept of Fagin-Williams share. The FTA problem [59] aims to minimize the payoff difference among workers using a game-theoretic approach. The OSM-KIID problem [58] aims to maximize the expected total profits and minimize the expected total number of blocking pairs, which should be zero to ensure stability in the stable marriage problem [19]. Another related work [31] extents the egalitarian stable marriage problem using a function of the waiting time as the preference score on task assignment in spatial crowdsourcing. However, none of the above solutions returns a set of best stable matchings by egalitarian cost.

Stable Marriage and its Variants. The *stable marriage problem* was introduced in 1962 [12] with a solution that finds a one-side-optimal stable marriage in $O(n^2)$. Since then, several measures of fairness have been proposed, including the *regret cost* [18], the *egalitarian cost* [24], and the *sex-equality cost* [30]. The *minimum-regret stable marriage* and the *egalitarian stable marriage* are found in polynomial time [18, 24], while minimizing the sex-equality cost is

NP-hard [27, 30, 38]. There are many other variants of SMP [19, 26]. Some extensions are about the nature of preference lists, such as *incomplete* preference lists [13], preference lists *with ties* [22], and both [25]. Relaxing the one-to-one constraint leads to the *Hospital/Residents problem* [19], which considers many-to-one matchings, and many-to-many stable marriages [37]. Relaxing the stability constraint, the *robust stable marriage problem* [15, 16] calls to find a marriage with a few unstable pairs. Although these problems have been studied for decades, little attention has been paid to *the k-best stable marriages* by egalitarian cost.

***k*-Best Enumeration Problems.** Problems calling to finding the k best solutions have been studied in many discrete optimization problems and applications for decades [8]; examples include the k shortest paths [7, 57], k-nearest-neighbor search in spatial database [41, 51], and k-best matchings in weighted graphs [4, 40]. A similar problem is to find the k-best assignments on a bipartite graph [40]. Chegireddy and Hamacher [4] give a $O(kn^3)$ algorithm that constructs a binary partition of the solution space. While such problems only consider symmetric or one-sided preferences, k-ESMP aims to find k stable matchings of high quality with asymmetric two-sided preferences. Besides, k-ESMP is **NP**-hard. Some k-best enumeration problems allow both the best solution and the k-best solutions to be found in polynomial time by exploiting optimal substructures [8]. Others, such as the k-minimum spanning tree problem [4], become more challenging when seeking more than one best solution ($k > 1$). k-ESMP belongs to the latter category.

Stable Marriage Enumeration. The number of stable marriages in a problem instance may grow exponentially in the input size [23] and is large in real-world markets [3, 20]; counting them is #**P**-hard [23, 42]. McVitie [39] provides an enumeration algorithm with time complexity $O(n^3N)$ in the worst case and $O(n^3N/\log^2 N)$ in the best case. Knuth [33, 34] proposes a similar enumeration method in the same time complexity. The set of stable marriages forms a *lattice*, while a *rotation poset* helps generate them [19]. Based on this idea, the enumeration algorithm introduced in [18] uses only $O(n^2 + nN)$ time and $O(n^2)$ space to enumerate all stable marriages by successively finding each stable marriage in $O(n)$ time. We revisit this classic enumeration algorithm in Section 4.

Solutions to stable marriage problem variants use agent-based procedures [17, 50], linear programming [46], and local search [14, 49, 54, 55]. Yet none of those finds *the k-best stable marriages by egalitarian cost* while traversing the associated lattice. We solve this **NP**-hard problem in a practicable manner using the rotation poset and novel pruning techniques.

3 PRELIMINARIES

Here we introduce fundamental concepts that aid the understanding of the problem and its solution.

3.1 SMP and ESMP

Stable Marriage Problem (SMP). An instance of SMP, $I = (X, Y, P)$, consists of a set of n agents on one side $X = \{x_1, ..., x_n\}$ and a set of n agents on another side $Y = \{y_1, ..., y_n\}$, where each agent (x or y) has a preference list on those on the other side, denoted as P_x or P_y; $P_y(x)$ ($P_x(y)$) is the position of x (y) in y's (x's) preference list. Table 1 shows the preference lists in our example for subsequent use. For example, $P_{x_1}(y_1) = 2$. A marriage is a matching that comprises n disjoint (x, y) couples, denoted as $\mathcal{M}$. We denote the mapping relationship as $\mathcal{M}(x) = y$ and $\mathcal{M}(y) = x$ if and only if $(x, y) \in \mathcal{M}$. Given a $\mathcal{M}$, if there is an agent x (e.g, a man) and an agent y (e.g, a woman) such that x prefers y to $\mathcal{M}(x)$ and y prefers x to $\mathcal{M}(y)$, (x, y) is a *blocking pair*. A marriage $\mathcal{M}$ without blocking pair is a *stable marriage*.

Given an instance of SMP, I, the Gale-Shapley (GS) algorithm [12] finds in $O(n^2)$ time either an *X-optimal* stable marriage $\mathcal{M}_X$, in which each x gets the most preferred mate and each y the least preferred mate they can get among stable marriages, or a *Y-optimal* stable marriage $\mathcal{M}_Y$, which reverses roles [39]. Both $\mathcal{M}_X$ and $\mathcal{M}_Y$ are biased in favor of one side and against the other side [49]. Table 3 shows the X-optimal stable marriage $\mathcal{M}_X$ in our example.

Egalitarian Stable Marriage Problem. The Egalitarian Stable Marriage (ESMP) [24] maximizes the cumulative happiness of all agents. Formally, the egalitarian cost of a stable marriage $\mathcal{M}$ is the sum of the positions of all agents in their partners' preference lists:

$$C(\mathcal{M}) = \sum_{(x,y)\in\mathcal{M}} P_x(y) + P_y(x) \tag{1}$$

ESMP seeks a stable marriage $\mathcal{M}$, also denoted as $\mathcal{M}^1$, with minimum egalitarian cost $C(\mathcal{M})$; it is solved in $O(n^4)$ time by a network-flow-based method which we introduce in the following.

3.2 All Stable Marriages, $\mathbb{M}$

Given a stable marriage instance I, we denote the set of all stable marriages of I as $\mathbb{M} = \{\mathcal{M}_1, \ldots, \mathcal{M}_N\}$, where N is its cardinality. Both the X-optimal and Y-optimal marriages are included in $\mathbb{M}$. Irving and Leather [23] studied a family of stable marriage instances which reveals that N grows exponentially in the input size n, hence it becomes intractable to explicitly store and search the set of all stable marriages.

Instead, we explore all stable marriages implicitly using a compact representational primitive called *rotation* [19, 21, 23]. A rotation r is *exposed* in a stable marriage $\mathcal{M}$ as an ordered list of pairs such that breaking each pair $(x_\rho, y_\rho) \in \mathcal{M}$ and re-coupling x_ρ with $y_{\rho+1}$ in a cyclic cascade shifts the matchings from r to $\tilde{r}$ constituting a new *stable* marriage $\mathcal{M}'$:

$$r = \{(x_\rho, y_\rho), (x_{\rho+1}, y_{\rho+1}), \ldots, (x_z, y_z)\}$$
$$\tilde{r} = \{(x_\rho, y_{\rho+1}), (x_{\rho+1}, y_{\rho+2}), \ldots, (x_z, y_\rho)\}$$

We call this process *rotation elimination*, denoted as $\mathcal{M}/r \rightarrow \mathcal{M}'$. Table 2 provides a complete list of rotations for our example instance.

Example 2 (Rotation and Rotation Elimination). *In Tables 2 and 3, rotation $r_2 = \{(x_3, y_7), (x_5, y_4), (x_8, y_2)\}$ is exposed in $\mathcal{M}_X$. $\mathcal{M}_X/r_2$ leads to a new stable marriage, where r_2 is eliminated to $\tilde{r}_2$, marked with wavy lines.*

A *partial order relationship* $r_i \prec r_j$ among rotations indicates that rotation r_i must be eliminated before rotation r_j, otherwise r_j cannot be exposed. For example, eliminating r_1 and r_2 from $\mathcal{M}_X$, in any order, leads to a new stable marriage $\mathcal{M}$, in which we find rotation r_4. A set of such relationships defines a *rotation poset* $R = (r, \prec)$. Given

a stable marriage instance I, we can construct[1] its *rotation poset* in $O(n^2)$ and represent it as a directed acyclic graph $G = (V, E_{\prec})$, where each node stands for a rotation r and a directed edge (r_i, r_j) for a $\prec$ relation between rotation r_i and rotation r_j. We denote the predecessors and successors of r as $Pred(r)$ and $Succ(r)$, respectively. For instance in Figure 3, $Prec(r_7) = \{r_1, r_2, r_3, r_4, r_5\}$ and $Succ(r_7) = \{r_9, r_{10}\}$.

Table 1: Preference Lists

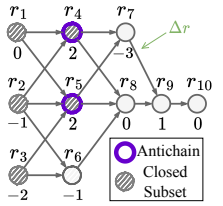
Preference Lists of X (ranked by distance)		Preference Lists of Y (ranked by utility)	
P_{x_1}	$y_3\ y_1\ y_5\ y_7\ y_4\ y_2\ y_8\ y_6$	P_{y_1}	$x_4\ x_3\ x_8\ x_1\ x_2\ x_5\ x_7\ x_6$
P_{x_2}	$y_6\ y_1\ y_3\ y_4\ y_8\ y_7\ y_5\ y_2$	P_{y_2}	$x_3\ x_7\ x_5\ x_8\ x_6\ x_4\ x_1\ x_2$
P_{x_3}	$y_7\ y_4\ y_3\ y_6\ y_5\ y_1\ y_2\ y_8$	P_{y_3}	$x_7\ x_5\ x_8\ x_3\ x_6\ x_2\ x_1\ x_4$
P_{x_4}	$y_5\ y_3\ y_8\ y_2\ y_6\ y_1\ y_4\ y_7$	P_{y_4}	$x_6\ x_4\ x_2\ x_7\ x_3\ x_1\ x_5\ x_8$
P_{x_5}	$y_4\ y_1\ y_2\ y_8\ y_7\ y_3\ y_6\ y_5$	P_{y_5}	$x_8\ x_7\ x_1\ x_5\ x_6\ x_4\ x_3\ x_2$
P_{x_6}	$y_6\ y_2\ y_5\ y_7\ y_8\ y_4\ y_3\ y_1$	P_{y_6}	$x_5\ x_4\ x_7\ x_6\ x_2\ x_8\ x_3\ x_1$
P_{x_7}	$y_7\ y_8\ y_1\ y_6\ y_2\ y_3\ y_4\ y_5$	P_{y_7}	$x_1\ x_4\ x_5\ x_6\ x_2\ x_8\ x_3\ x_7$
P_{x_8}	$y_2\ y_6\ y_7\ y_1\ y_8\ y_3\ y_4\ y_5$	P_{y_8}	$x_2\ x_5\ x_4\ x_3\ x_7\ x_8\ x_1\ x_6$

Table 2: Rotations

r	After Elimination, $\tilde{r}$	Δr	*Parent*
$r_1 = (x_1, y_3)(x_2, y_1)$	$(x_1, y_1)(x_2, y_3)$	0	
$r_2 = (x_3, y_7)(x_5, y_4)(x_8, y_2)$	$(x_3, y_4)(x_5, y_2)(x_8, y_7)$	-1	
$r_3 = (x_4, y_5)(x_7, y_8)(x_6, y_6)$	$(x_4, y_8)(x_7, y_6)(x_6, y_5)$	-2	
$r_4 = (x_2, y_3)(x_3, y_4)$	$(x_2, y_4)(x_3, y_3)$	2	r_1, r_2
$r_5 = (x_1, y_1)(x_6, y_5)(x_8, y_7)$	$(x_1, y_5)(x_6, y_7)(x_8, y_1)$	2	r_1, r_2, r_3
$r_6 = (x_4, y_8)(x_7, y_6)(x_5, y_2)$	$(x_4, y_6)(x_7, y_2)(x_5, y_8)$	-1	r_2, r_3
$r_7 = (x_3, y_3)(x_8, y_1)$	$(x_3, y_1)(x_8, y_3)$	-3	r_4, r_5
$r_8 = (x_2, y_4)(x_5, y_8)(x_6, y_7)$	$(x_2, y_8)(x_5, y_7)(x_6, y_4)$	0	r_4, r_5, r_6
$r_9 = (x_1, y_5)(x_5, y_7)(x_8, y_3)$	$(x_1, y_7)(x_5, y_3)(x_8, y_5)$	1	r_7, r_8
$r_{10} = (x_3, y_1)(x_7, y_2)(x_5, y_3)(x_4, y_6)$	$(x_3, y_2)(x_7, y_3)(x_5, y_6)(x_4, y_1)$	0	r_9

Table 3: Stable Marriage Examples for Rotation Elimination

$\mathcal{M}$	$=$	$C(\mathcal{M})$
$\mathcal{M}_X$	$(x_1, y_3), (x_2, y_1), (x_3, y_7), (x_4, y_5)$ $(x_5, y_4), (x_6, y_6), (x_7, y_8), (x_8, y_2)$	55
$\mathcal{M}_X/r_2$	$(x_1, y_3), (x_2, y_1), (x_3, y_4), (x_4, y_5)$ $(x_5, y_2), (x_6, y_6), (x_7, y_8), (x_8, y_7)$	56
$\mathcal{M}_X/\mathcal{S}$	$(x_1, y_5), (x_2, y_4), (x_3, y_3), (x_4, y_8)$ $(x_5, y_2), (x_6, y_7), (x_7, y_6), (x_8, y_1)$	54
$\mathcal{M}_X/R \to \mathcal{M}_Y$	$(x_1, y_7), (x_2, y_8), (x_3, y_2), (x_4, y_1)$ $(x_5, y_6), (x_6, y_4), (x_7, y_3), (x_8, y_5)$	57



Example:
$\mathcal{A} = \{r_4, r_5\}$
$\mathcal{S} = \{r_1, r_2, r_3, r_4, r_5\}$

Figure 3: The corresponding directed acyclic graph G.

A subset $\mathcal{A} \subseteq R$ is an *antichain* of R if all rotations in $\mathcal{A}$ are incomparable by any partial order, i.e., $\forall r_i, r_j \in \mathcal{A}, r_i \notin Prec(r_j) \wedge r_i \notin Succ(r_j)$. A subset $\mathcal{S} \subseteq R$ is a *closed subset* of R if $\mathcal{S}$ contains an antichain $\mathcal{A}$ and all its predecessors in R. By Theorem 1, eliminating a closed subset $\mathcal{S}$ generates a corresponding stable marriage $\mathcal{M}$.

Theorem 1. *[23] Given a stable marriage instance I, there is a one-to-one correspondence among the set of stable marriages $\mathbb{M}$, the set of closed subsets $\mathbb{S}$, and the set of antichains $\mathbb{A}$.*

[1]The interested reader may refer to [19, 23, 24] for details.

Given the X-optimal stable marriage $\mathcal{M}_X$, we may generate every other stable marriage $\mathcal{M}$ by eliminating all rotations in the closed subset $\mathcal{S}$ that contains its corresponding antichain $\mathcal{A}$. We denote this operation as $\mathcal{M}_X/\mathcal{S} \to \mathcal{M}$. Further, we get $\mathcal{M}_Y$ by eliminating all rotations in R.

Example 3 (Antichain, Closed subset and Stable Marriage). *As Figure 3 shows, $\mathcal{A} = \{r_4, r_5\}$ is an antichain, as there is no order among the elements. The antichain $\mathcal{A}$ and its predecessors, $\{r_1, r_2, r_3\}$, derive the closed subset $\mathcal{S} = \{r_1, r_2, r_3, r_4, r_5\}$. Table 3 shows the corresponding stable marriage $\mathcal{M}_X/\mathcal{S}$.*

Given an antichain $\mathcal{A}$, we find the corresponding closed subset $\mathcal{S}$ by breadth-first search in G and generate the corresponding matching stable marriage $\mathcal{M}$ by eliminating rotations [23] in $O(n^2)$, shown in Equation 2. Thus, if we enumerate all antichains $\mathcal{A} \in \mathbb{A}$, we generate all stable marriages $\mathcal{M} \in \mathbb{M}$; still, counting antichains is #**P**-hard [23, 42].

$$\mathcal{A} \xrightarrow{\text{BFS on } G} \mathcal{S} \xrightarrow{\text{rotation elimination}} \mathcal{M} \tag{2}$$

3.3 Solving ESMP

ESMP is easier than counting all stable marriages, as it only calls to find a stable marriage having the lowest egalitarian cost. To calculate the egalitarian cost $C(\mathcal{M})$ of stable marriage $\mathcal{M}$ generated by $\mathcal{M}_X/\mathcal{S} \to \mathcal{M}$, we keep track of the egalitarian cost changes incurred by eliminating the rotations in $\mathcal{S}$. Equation 1 is rewritten as:

$$C(\mathcal{M}) = C(\mathcal{M}_X) - \sum_{r_i \in \mathcal{S}} \Delta r_i \tag{3}$$

where Δr_i is the *cumulative difference* of preferences between rotation r_i and its outcome position $\tilde{r}_i$:

$$\Delta r_i = \sum_{(x_\rho, y_\rho) \in r_i} \left[P_{x_\rho}(y_\rho) + P_{y_\rho}(x_\rho) \right] - \sum_{(x_\rho, y_{\rho+1}) \in \tilde{r}_i} \left[P_{x_\rho}(y_{\rho+1}) + P_{y_{\rho+1}}(x_\rho) \right] \tag{4}$$

According to Equation 3, minimizing egalitarian cost amounts to maximizing $\sum_{r_i \in \mathcal{S}} \Delta r_i$, denoted as $\Delta\mathcal{S}$ in the subsequent discussion. As $C(\mathcal{M}_X)$ is calculated by the Gale-Shapley algorithm, we can minimize $C(\mathcal{M})$ by maximizing $\Delta\mathcal{S}$. ESMP is then reduced to finding **a maximum-weight closed subset** $\mathcal{S}_{\max}$ in the rotation poset R, such that $\Delta\mathcal{S}_{max}$ is maximized. We can do so by a network flow algorithm, Mincut, which finds $\mathcal{S}_{\max}$ on G in $O(n^4)$ time [24].

Example 4 (The maximum-weight closed subset and $\mathcal{M}^1$). *Applying the Mincut algorithm on G, we find the maximum-weight closed subset $\mathcal{S} = \{r_1, r_2, r_3, r_4, r_5\}$ with $\Delta\mathcal{S} = 0 - 1 - 2 + 2 + 2 = 1$. In our example, it is $\mathcal{M}_X/\mathcal{S} \to \mathcal{M}^1$ with $C(\mathcal{M}^1) = 55 - 1 = 54$.*

Table 4 gathers the quantities and complexities we have introduced. In summary, to address the ESMP problem, we may construct the rotation poset and subsequently find $\mathcal{S}_{\max}$ within it, with a total time complexity of $O(n^4)$.

Table 4: Quantities and Complexities [12, 19, 23, 24]

Quantity	Value	Algorithm	Complexity
Instance Size	n	Find $\mathcal{M}_X$ or $\mathcal{M}_Y$ (Gale-Shapley)	$O(n^2)$
Stable Marriage	N	Find all rotations	$O(n^2)$
Rotations (V_R)	$O(n^2)$	Find all $E_\prec$	$O(n^2)$
$E_\prec$	$O(n^2)$	Find $\mathcal{M}^1$ (Mincut)	$O(n^4)$

4 THE k-BEST EGALITARIAN STABLE MARRIAGE PROBLEM

The k-best Egalitarian Stable Marriage Problem (k-ESMP) can be formally defined below:

Definition 1 (k-ESMP). *Given a stable marriage instance $I = (X, Y, P)$, let the set of all stable marriages be $\mathbb{M} = \{\mathcal{M}_1, \ldots, \mathcal{M}_N\}$; the* k-ESMP *finds k ($1 < k \leq N$) stable marriages $\mathbb{M}(k) = \{\mathcal{M}^1, \ldots, \mathcal{M}^k\}$ such that $\mathcal{M}(k) \subseteq \mathbb{M}$ and $\forall \mathcal{M}' \in \mathbb{M} \setminus \mathbb{M}(k)$, $C(\mathcal{M}^1) \leq \ldots \leq C(\mathcal{M}^k) \leq C(\mathcal{M}')$, where C is the egalitarian cost.*

We define $k > 1$ in Definition 1 to differentiate the problem from ESMP, which can be efficiently solved in $O(n^4)$ time, as discussed in Section 3.3.

Theorem 2. k-ESMP *is* **NP**-*hard.*

Proof. There are at most $n!$ stable marriages. Assume a polynomial time algorithm T identifies the kth stable marriage by egalitarian cost $C(\mathcal{M}^k)$, if such exists. We could use T to count all stable marriages via binary search, i.e., find $\mathcal{M}^k$, $\mathcal{M}^{2k}$, $\mathcal{M}^{4k}$, …, until we get N, in time $O(\log n!) = O(n \log n)$. This result contradicts the fact that counting stable marriages is #**P**-hard [19, 42]. □

By Theorem 2, we cannot solve k-ESMP in polynomial time. A brute-force solution is to enumerate all stable marriages, find and sort a set of k having the best egalitarian costs, maintaining $\mathbb{M}(k)$ by a max-heap structure of size k. We build on this idea by applying two lower bounds, namely the *layer bound* and the *rotation bound*, to reduce the exploration space in stable-marriage enumeration. To our knowledge, this is the first work to exploit partial order relationships and apply bounding techniques on enumerating egalitarian stable marriages.

4.1 Layer Lower Bound, LB

Since k-ESMP calls to find the k-best stable marriages by egalitarian cost, we can prune from the enumeration process stable marriages having large egalitarian costs given a *lower bound* for the unrevealed exploration space. We carefully exploit the topological structure of the rotation graph G (cf. Figure 3) and propose two bounding techniques, the *layer bound* **LB** and *rotation bound* **RB**. Table 5 summarizes some key notations used in the following discussion.

We first define the *layer level* of a rotation (vertex) r in G.

Definition 2 (Layer Level, $r.\ell$). *The* layer level *$r.\ell$ of a rotation r in G is the longest-path length from the root of G (i.e., the X-optimal matching) to r.*

In Figure 4, $r_4.\ell$ is 2 and $r_{10}.\ell$ is 5. Based on the rotation layer level $r.\ell$, we define the layer rotation subgraph G^ℓ as:

Definition 3 (Layer Subgraph, G^ℓ). *A* layer subgraph *$G^\ell = (V^\ell, E^\ell)$ is an induced subgraph of G, where $V^\ell = \{r \in V | r.\ell \leq \ell\}$ and $E^\ell = \{(u, v) \in E_\prec | u.\ell, v.\ell \leq \ell\}$.*

Table 5: Key Notations of $\mathbf{LB}_\ell$ and $\mathbf{RB}_r$

Notation	Description
$\mathbf{LB}_\ell$	the layer lower bound of layer ℓ calculated by Eq 5
G^ℓ	the layer-ℓ subgraph (Def 3)
$\mathbb{M}(G^\ell)$	stable marriages derived from G^ℓ (see Example 5)
$\mathcal{S}_L^\ell$	the maximum-weight closed subset on G^ℓ
$\mathcal{M}_L^\ell$	the stable marriage with the minimum egalitarian cost in $\mathbb{M}(G^\ell)$, i.e., $\mathcal{M}_X / \mathcal{S}_L^\ell \rightarrow \mathcal{M}_L^\ell$
$\mathbf{RB}_r$	the rotation lower bound of r calculated by Eq 7
$R^+, R^{\geq +}$	the rotation sets such that $\Delta r > 0$, and $\Delta r \geq 0$
$Prec^-(r)$	all negative rotations in the predcessors of r
$Succ^+(r)$	all positive rotations in the successors of r

For simplicity, we use $\mathbb{M}(G^\ell)$ to represent the set of stable marriages generated from the layer subgraph, i.e, $\mathbb{M}(G^\ell) = \{\mathcal{M}_\mathcal{S} | \forall \mathcal{S} \subseteq G^\ell\}$. We can find the stable marriage $\mathcal{M}_L^\ell$ with the minimum egalitarian cost in $\mathbb{M}(G^\ell)$, by applying the Mincut algorithm (see Section 3.3) on the layer subgraph G^ℓ to find the maximum-weight closed subset $\mathcal{S}_L^\ell$ in G^ℓ.

Therefore, the egalitarian cost of $\mathcal{M}_L^\ell$, $C(\mathcal{M}_L^\ell)$, *lower-bounds* the egalitarian cost of stable marriages in $\mathbb{M}(G^\ell)$. We calculate the lower bound for a layer ℓ as follows.

$$\mathbf{LB}_\ell = C(\mathcal{M}_L^\ell) = C(\mathcal{M}_X) - \Delta\mathcal{S}_L^\ell \tag{5}$$

Lemma 1 (Layer Bound Monotonicity). *The layer bound $\mathbf{LB}_\ell$ is a non-increasing function of layer ℓ.*

Proof. We prove this by contradiction. Assume there exists a case where $\mathbf{LB}_\ell > \mathbf{LB}_{\ell-1}$, with the corresponding closed subsets of $\mathbf{LB}_\ell$ and $\mathbf{LB}_{\ell-1}$ being $\mathcal{S}_L^\ell$ and $\mathcal{S}_L^{\ell-1}$, respectively. By construction, $\mathcal{S}_L^\ell$ and $\mathcal{S}_L^{\ell-1}$ are the best, in terms of egalitarian cost, closed subsets of G^ℓ and $G^{\ell-1}$, respectively. Since $G^{\ell-1} \subseteq G^\ell$, $\mathbf{LB}_\ell$ is at most equal to $\mathbf{LB}_{\ell-1}$, a contradiction. □

With the aid of Lemma 1, we apply the following pruning rule as we scan layers *from right to left*.

Pruning Rule 1. *If the layer lower bound $\mathbf{LB}_\ell$ is* not lower *than the running result at position k in max heap $\mathbb{M}(k)$, i.e., if $\mathbf{LB}_\ell \geq C(\mathbb{M}(k).top())$, then we prune the stable marriages at G^ℓ, i.e., $\mathbb{M}(G^\ell)$.*

Example 5 (Layer Lower Bound). *By Definition 2, G^1 contains the first rotation layer, $\{r_1, r_2, r_3\}$ and G^2 contains $\{r_1 - r_6\}$ in Figure 4. As we can infer from Table 6, it is $\mathbb{M}(G^1) = \{\mathcal{M}_2 - \mathcal{M}_8\}$ and $\mathbb{M}(G^2) = \{\mathcal{M}_2 - \mathcal{M}_{17}\}$. Assume $k = 3$ and the running 3-best results are $\mathbb{M}(k) = \{\mathcal{M}_{14}, \mathcal{M}_9, \mathcal{M}_{16}\}$. Suppose the best stable marriage found by the Mincut algorithm in G^1 is $\mathcal{M}_L^1 = \mathcal{M}_2$. If $C(\mathcal{M}_L^1) \geq C(\mathbb{M}(k).top())$, we prune the stable marriage enumeration at G^1 by Pruning Rule 1, i.e., we prune $\mathcal{M}_2 - \mathcal{M}_8$.*

4.2 Rotation Lower Bound, RB

Due to Equations (3) and (4), the egalitarian cost of *any* stable marriage is lower-bounded by subtracting the cumulative differences of all positive rotations $R^+ \subseteq R$ from the cost of the X-optimal stable marriage:

$$\Omega = C(\mathcal{M}_X) - \sum_{r_i \in R^+} \Delta r_i \leq C(\mathcal{M}) \tag{6}$$

Table 6: All Closed Subsets and Antichains (antichains are marked with underline.)

Closed Subsets (i.e., $\mathcal{M}_X/\mathcal{S}_i \to \mathcal{M}_i$)		
1. $\emptyset$ (i.e., $\mathcal{M}_X$)	9. $\{r_1, r_2, \underline{r_4}\}$	17. $\{r_1, r_2, r_3, \underline{r_4, r_5, r_6}\}$
2. $\{\underline{r_1}\}$	10. $\{r_1, r_2, r_3, \underline{r_5}\}$	18. $\{r_1 - r_5, \underline{r_7}\}$
3. $\{\underline{r_2}\}$	11. $\{r_2, r_3, \underline{r_6}\}$	19. $\{r_1 - r_5, \underline{r_6, r_7}\}$
4. $\{\underline{r_3}\}$	12. $\{\underline{r_1}, r_2, r_3, \underline{r_6}\}$	20. $\{r_1 - r_6, \underline{r_7, r_8}\}$
5. $\{\underline{r_1, r_2}\}$	13. $\{r_1, r_2, \underline{r_3, r_4}\}$	21. $\{r_1 - r_6, \underline{r_8}\}$
6. $\{\underline{r_1, r_3}\}$	14. $\{r_1, r_2, r_3, \underline{r_4, r_5}\}$	22. $\{r_1 - r_8, \underline{r_9}\}$
7. $\{\underline{r_2, r_3}\}$	15. $\{r_1, r_2, r_3, \underline{r_4, r_6}\}$	23. $\{r_1 - r_9, \underline{r_{10}}\}$ (i.e., $\mathcal{M}_Y$)
8. $\{\underline{r_1, r_2, r_3}\}$	16. $\{r_1, r_2, r_3, \underline{r_5, r_6}\}$	

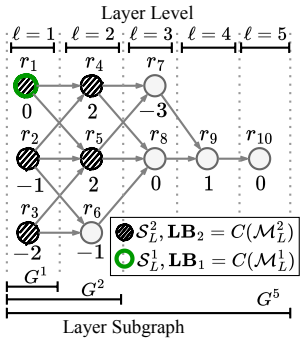


Figure 4: Layer Bound, $\mathbf{LB}_1$ and $\mathbf{LB}_2$

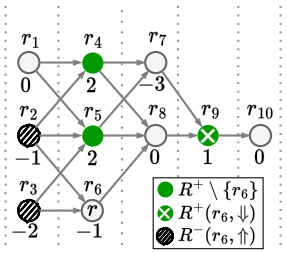


Figure 5: Rotation Bound, $\mathbf{RB}_{r_6}$ $\mathbf{RB}_{r_6} = C(\mathcal{M}_X) - (\Delta r_2 + \Delta r_3 + \Delta r_4 + \Delta r_5 + \Delta r_6) = 55 - 0 = 55$

However, the bound in Equation (6) is too loose, as it is valid for any stable marriage. Given a rotation r, we derive a *tighter* rotation lower bound $\mathbf{RB}_r$ for stable marriages derived from any antichain that contains r, based on the successors and predecessors of r in G, as follows:

$$\mathbf{RB}_r = C(\mathcal{M}_X) - \left(\Delta r + \sum_{r_i \in R^+ \setminus \{r\}} \Delta r_i + \sum_{r_j \in Prec^-(r)} \Delta r_j - \sum_{r_z \in Succ^+(r)} \Delta r_z \right) \tag{7}$$

Compared to Equation (6), the rotation lower bound in Equation (7) has two additional terms. First, we *subtract* the cumulative differences of all negative rotations in the predecessor set of r, $Prec^-(r)$, since predecessors of r are in any closed subset whose corresponding antichain contains r by the definition of closed subset, hence necessarily enacted in any matching derived from an antichain that contains r. In Figure 5, the cumulative differences of r_2 and r_3 are subtracted in the bound for r_6. Second, we *do not* subtract the cumulative differences of all positive rotations in the successor set of r, $Succ^+(r)$, since successors of r cannot be in an antichain that contains r. In Figure 5, we exclude r_9 from subtractions in the bound for r_6. Further, r should be in the antichain, no matter if positive or negative.

We exploit the correctness of $\mathbf{RB}_r$ as below:

Lemma 2 (Correctness of Rotation Bound). *For any stable marriage $\mathcal{M}$ corresponding to antichain $\mathcal{A}$, and rotation $r \in \mathcal{A}$, it holds that $\mathbf{RB}_r \leq C(\mathcal{M})$.*

Proof. By construction, $C(\mathcal{M}) \geq \Omega + \sum_{r_z \in Succ^+(r)} \Delta r_x$, since the rotations in $Succ^+(r)$ are not exposed in $\mathcal{M}$. Yet due to Equations (6) and (7), $\mathbf{RB}_r \leq \Omega + \sum_{r_z \in Succ^+(r)}$; it follows that $\mathbf{RB}_r \leq C(\mathcal{M})$. □

We define our second pruning rule based on $\mathbf{RB}_r$ as follows.

Pruning Rule 2. *If the rotation bound of r, $\mathbf{RB}_r$, is* not lower *than the running result at position k in $\mathbb{M}(k)$, i.e., $\mathbf{RB}_r \geq C(\mathbb{M}(k).top())$, then we prune the stable marriages whose corresponding antichains contain r.*

Example 6 (Rotation Lower Bound). *For rotation r_6, we can calculate $\mathbf{RB}_{r_6}$ by Equation (7), as Figure 5 shows. $\mathbf{RB}_{r_6}$ lower-bounds the costs of stable marriages $\mathcal{M}_{11}, \mathcal{M}_{12}, \mathcal{M}_{15}, \mathcal{M}_{16}, \mathcal{M}_{17}$ and $\mathcal{M}_{19}$, which can be pruned if $\mathbf{RB}_{r_6} \geq C(\mathbb{M}(k).top())$. Note that, even though r_6 is in closed subsets $\mathcal{S}_{20} - \mathcal{S}_{23}$, it is not in their corresponding antichains, hence the cost of the corresponding stable marriages $\mathcal{M}_{20} - \mathcal{M}_{23}$ is not bounded by $\mathbf{RB}_{r_6}$.*

4.3 ENUM* Enumeration Algorithm

Unfortunately, the two lower bounds we have proposed cannot work with the classic enumeration algorithm, ENUM (Section 2). As we saw in Lemma 1 and Example 5 in Section 4.1, the layer bound is non-increasing with layer ℓ, following a breadth-first enumeration from right to left on the rotation poset. However, ENUM enumerates stable marriages via depth-first search, exploring all closed subsets of a given rotation r before moving on. The rotation bound of r may not suffice to prune these subsets early on, as the k-best result $\mathbb{M}(k).top()$ may still be far from the optimal result. To utilize **LB** and **RB** fruitfully, we introduce a new, breadth-first-search enumeration algorithm, ENUM*, and apply the two bounds on it to yield ENUM*-LR, a pruning-intensive algorithm that finds the k-best stable marriages.

ENUM*. As discussed in Section 3.2, we can generate a stable marriage $\mathcal{M}$ via its corresponding antichain $\mathcal{A}$ and closed subset $\mathcal{S}$ by Equation 2. Thereby, we can enumerate stable marriages by *antichain enumeration*. ENUM enumerates all stable marriages by expanding a closed subset with an exposed rotation in each step. In a similar way, we expand an antichain with an incomparable rotation. By the definition of antichains, the enumeration process recursively *expands* an antichain $\mathcal{A}$ with a new rotation r_{new} from among the *incomparable* rotations $R_{\mathcal{A}}$ (for $\mathcal{A}$), to get a new antichain $\mathcal{A} \cup r_{new}$. Further, we exploit the incomparability property of antichains (Corollary 1) to avoid redundant checks; this analysis resembles that of lattice theory [10, 19].

Corollary 1 (Incomparability). *Given an antichain $\mathcal{A}$ and rotation r' such that $\exists r \in \mathcal{A}, r' \in Succ(r)$, the enumeration process expanding $\mathcal{A}$ excludes r'.*

Due to Corollary 1, we exclude rotations comparable to r, $R_r^{\times} = Succ(r) \cup r$, from the process expanding antichain $\mathcal{A}$, for all rotations $r \in \mathcal{A}$; thus, the incomparable rotations to be used for expanding an antichain $\mathcal{A}$ are:

$$R_{\mathcal{A}} = R \setminus \cup_{r \in \mathcal{A}} R_r^{\times}$$

We add each $r_{new} \in R_{\mathcal{A}}$ to $\mathcal{A}$ to enumerate a new antichain $\mathcal{A} \cup r_{new}$, and update the incomparable rotations for expanding the new

antichain $\mathcal{A} \cup r_{new}$.

$$R_{\mathcal{A} \cup r_{new}} = R_{\mathcal{A}} \setminus R^{\times}_{r_{new}}$$

We enumerate all antichains by this recursive mechanism.

Algorithm 1 ENUM*

Input: $\mathcal{M}_X, R, k$
Output: $\mathbb{M}(k)$
1: Max heap $\mathbb{M}(k) \leftarrow \emptyset$
2: EnumAntichain($\emptyset, R, \mathbb{M}(k)$)
3: **return** $\mathbb{M}(k)$
4: **function** EnumAntichain($\mathcal{A}, R_{\mathcal{A}}, \mathbb{M}(k)$)
5: $\mathcal{A} \rightarrow \mathcal{S} \rightarrow \mathcal{M}$ ▷ Generate matching using Eq. 2
6: UpdateKESM($\mathcal{M}, \mathbb{M}(k)$) ▷ Update the k best solutions
7: **for** $r_{new} \in R_{\mathcal{A}}$, in descending order of ℓ **do**
8: $R_{\mathcal{A} \cup r_{new}} = R_{\mathcal{A}} \setminus R^{\times}_{r_{new}} \setminus \{r \in R_{\mathcal{A}} | r <_{id} r_{new}\}$
9: EnumAntichain($\mathcal{A} \cup r_{new}, R_{\mathcal{A} \cup r_{new}}, \mathbb{M}(k)$)
10: **function** UpdateKESM($\mathcal{M}, \mathbb{M}(k)$)
11: **if** $C(\mathcal{M}) < C(\mathbb{M}(k).top())$ **then** ▷ C is computed by Eq. 3
12: $\mathbb{M}(k)$.pop(); $\mathbb{M}(k)$.push($\mathcal{M}$)

Algorithm 1 shows the pseudocode of our baseline enumeration ENUM*. A recursive mechanism (Lines 4 and 9) enumerates feasible antichains. A running antichain $\mathcal{A}$ picks any feasible rotation r_{new} from the incomparable rotation set $R_{\mathcal{A}}$ (Line 7), initially set to R (Line 1). To enumerate the next antichain, we exclude $R^{\times}_{r_{new}}$ from $R_{\mathcal{A}}$. Redundant enumerations may occur in case two different rotations are to be enumerated by each other's antichains. To avoid such redundancy, we enumerate rotations in a strict order (e.g., by vertex id), excluding from the process any rotation whose order is smaller than the newly added rotation r_{new} (Line 8). We tabulate each computed $Succ(r)$ to avoid redundant re-computations thereof in Line 8. We update the result set if the cost of $\mathcal{M}$ is not worse than the running stable marriage at position k (Lines 6 and 12). Since we maintain $\mathbb{M}(k)$ by a heap structure, it can return the ranking result by extracting the root element until empty.

Example 7 (ENUM*). *Assume the current antichain is $\mathcal{A} = \emptyset$ and $R_{\mathcal{A}} = R$. In Line 7 we pick an incomparable rotation r_5 in $R_{\mathcal{A}}$ after we have completed the loops of $r_{10}, r_9, \ldots, r_6$). We then update $\mathcal{A} \cup r_{new} = \{r_5\}$, $R_{\mathcal{A} \cup r_{new}} = R \setminus \{r_1 - r_5, r_7 - r_{10}\} = \{r_6\}$, where $\{r \in R_{\mathcal{A}} | r <_{id} r_5\} = \{r_1, r_2, r_3, r_4\}$. We remove rotations with smaller indexes to avoid repeated enumeration, e.g., if $\mathcal{A} = \{r_4\}$, then r_{new} can be r_5.*

Better Order. ENUM* enumerates all stable marriages without omission or duplication since each rotation and its incomparable rotations are enumerated in a strict order (from large index to small index). For layer bound, it postpones the enumeration of the small layer subgraphs. For rotation bound, it gives each rotation a chance to prune unpromising antichains. As an example, when we enumerate from r_6, we only enumerate $\mathcal{M}_{11}$ and $\mathcal{M}_{19}$. Other antichains that contain r_6 will be enumerated later. This will increase the pruning efficiency of the bound as the best-so-far result improves throughout the execution.

Complexity. The time complexity of ENUM* is $O(n^2 + \mathcal{U} \cdot \log k + Q \cdot N)$, where (i) $O(n^2)$ is to construct G^R; (ii) $O(\mathcal{U} \cdot \log k)$ is to maintain the max heap where $\mathcal{U}$ is the count of updates, as it takes $O(\log k)$ time to remove the top element and insert a better result into the heap for $\mathcal{U}$ times (Algorithm 1, Lines 11–12); in the worst case, $\mathcal{U}$ can increase up to N, where we enumerate all stable marriages from $\mathcal{M}^N, \mathcal{M}^{N-1}, \ldots, \mathcal{M}^1$, i.e., from the worst result to the best result; we also give the statistics of $\mathcal{U}$ in experiments, where $\mathcal{U} \ll N$ in practice; and (iii) $O(Q \cdot N)$ is to scan the $O(n^2)$ successors of each rotation in the running antichain, a cost we denote as Q for each stable marriage. In ENUM, the cost is $O(n^2 + \mathcal{U} \cdot \log k + n \cdot N)$, as it only requires $O(n)$ time for each stable marriage. Our objective is not to deploy ENUM* but rather to apply two lower bounds to reduce the search space N, which dominates the running time, as we describe in the next section.

4.4 Our solution, ENUM*-LR

Algorithm 2 shows our enhanced recursive solution, ENUM*-LR. Line 6 checks the layer lower bound before the rotation lower bound $\mathbf{RB}_r$ for rotation r newly added in antichain set $\mathcal{A}$, as the former is more likely to reuse bookkept results.

Algorithm 2 ENUM*-LR

Input: $\mathcal{M}_X, R, k$
Output: $\mathbb{M}(k)$
1: Max heap $\mathbb{M}(k) \leftarrow \emptyset$
2: EnumAntichain($\emptyset, R, null, \mathbb{M}(k)$)
3: **return** $\mathbb{M}(k)$
4: **function** EnumAntichain($\mathcal{A}, R_{\mathcal{A}}, r, \mathbb{M}(k)$)
5: $\ell = \max_{r' \in \mathcal{A} \cup R_{\mathcal{A}}} \{r'.\ell\}$ ▷ Max. level of possible antichains
6: **if** $\mathbf{LB}_\ell \geq C(\mathbb{M}(k).top())$ or $\mathbf{RB}_r \geq C(\mathbb{M}(k).top())$ **then**
7: **return** ▷ Check layer bound and rotation bound
8: $\mathcal{A} \rightarrow \mathcal{S} \rightarrow \mathcal{M}$ ▷ Generate matching using Eq. 2
9: UpdateKESM($\mathcal{M}, \mathbb{M}(k)$) ▷ Update the k best results(in Alg. 1)
10: **for** $r_{new} \in R_{\mathcal{A}}$, in descending order of ℓ **do**
11: $R_{\mathcal{A} \cup r_{new}} = R_{\mathcal{A}} \setminus R^{\times}_{r_{new}} \setminus \{r \in R_{\mathcal{A}} | r <_{id} r_{new}\}$
12: EnumAntichain($\mathcal{A} \cup r_{new}, R_{\mathcal{A} \cup r_{new}}, r_{new}, \mathbb{M}(k)$)

Implementation details. We calculate the layer bound $\mathbf{LB}_\ell$ by the Mincut algorithm (Section 3.3) on a layer subgraph G^ℓ in $O(n^4)$ time. In the worst case, the enumeration calls the Minuct algorithm L times, where L is the number of layers in the rotation graph G. We contain this number by a simple process. After we determine the best closed subset $\mathcal{S}^\ell_L$ of G^ℓ, we find the maximum layer level ℓ' of rotations in $\mathcal{S}^\ell_L$. A closed subset $\mathcal{S}^{\ell'}_L$ of $G^{\ell'}$ is identical to that of G^ℓ since $G^{\ell'} \subseteq G^\ell$. Thereby, we safely eschew calling the Mincut algorithm in *intermediate* layer subgraphs, $G^{\ell'}, G^{\ell'+1}, \ldots, G^{\ell-1}$. In terms of Figure 4, suppose $\mathcal{S} = \{r_1 - r_5\}$ is the maximum-weight closed subset found in G^4. Since the maximum layer level in $\mathcal{S}$ is 2, we eschew calling Mincut in G^2 and G^3. Lastly, we bookkeep the result of G^ℓ in the entire enumeration process.

We derive the rotation lower bound $\mathbf{RB}_r$ by iterating over $Prec^-(r)$ and $Succ^+(r)$ in $O(n^2)$ and bookkeep the resulting $\mathbf{RB}_r$ in $O(n^2)$ space. We prune unpromising rotations from the candidate set $R_{\mathcal{A}}$ by storing the set of computed $\mathbf{RB}_r$ values in a max-heap. When we update the k-best matching set $\mathbb{M}(k)$, we screen the max-heap and mark rotations as *unpromising* if their bound is not lower than the cost of the k-th best matching, i.e., if $\mathbf{RB}_r \geq C(\mathbb{M}(k).top())$; we

do not consider these *unpromising* rotations when constructing a new rotation candidate set $R_{\mathcal{A}}$.

Complexity. The time complexity of ENUM*-LR (applying both Pruning Rule 1 and Pruning Rule 2) is $O(n^4 + Q \cdot \mathcal{H}_{LR} + \mathcal{U} \cdot \log k)$. The $O(n^4)$ cost is to calculate layer bounds and rotation bounds. In practice, the bound calculation cost is much better than the worst case since the number of rotations is much smaller than n^2 (see Section 5.2). ENUM*-LR is to reduce the enumeration space from N to a pruned space $\mathcal{H}_{LR}$. For layer bound, the search space $\mathcal{H}_L$ is $N - \sum_{\alpha=1}^{\frac{n}{2}} \binom{|V^\ell|}{\alpha}$ in the best case, where α is the size of an antichain, which is at most the width of G. Assuming only one rotation bound works, the search space $\mathcal{H}_R$ in the best case is $N - \sum_{\alpha=1}^{n/2} \binom{|R|}{\alpha}$. The enumeration space can be reduced further with more unpromising rotations. In the experiments, the number of pruned rotations can be more than half of $|R|$. Even while the pruning spaces of $\mathbf{LB}_\ell$ and $\mathbf{RB}_r$ overlap, the search space of ENUM*-LR is smaller than that of ENUM*-L and ENUM*-R, which only apply **LB** or **RB**.

4.5 Heuristic yet Stable Solutions

Finding the exact k-best stable marriages to k-ESMP may still be intractable if the problem instance is hard, requiring the enumeration of an exponential number of stable marriages. To solve this problem, we propose two heuristic solutions, ENUM*-ϵ and ENUM*-LR-ϵ.

Table 7: The antichains without negative rotations are likely to be better solutions.

The 4-best closed subsets (the antichains are marked by underline.)
$\mathcal{S}_{14} = \{r_1, r_2, r_3, \underline{r_4, r_5}\}, C(\mathcal{M}_{14}) = 54$
$\mathcal{S}_9 = \{r_1, r_2, \underline{r_4}\}, C(\mathcal{M}_9) = 54$
$\mathcal{S}_{17} = \{r_1, r_2, r_3, \underline{r_4, r_5, r_6}\}, C(\mathcal{M}_{17}) = 55$
$\mathcal{S}_{21} = \{r_1, r_2, r_3, r_4, r_5, r_6, \underline{r_8}\}, C(\mathcal{M}_{21}) = 55$

ENUM*-ϵ. Our heuristic algorithm, ENUM*-ϵ, enumerates antichains by the same process as ENUM* (Alg. 1). To reduce the search space, we retain only non-negative rotations, $R^{\geq +}$, in the enumeration set $R_{\mathcal{A}}$ (Line 4 of Algorithm 1). Our rationale is that, if an antichain contains negative rotations, we can find an antichain preferable in terms of egalitarian cost by excluding negative rotations. For example, in Table 6, the corresponding antichain of $\mathcal{M}_{17}$ is $\{r_4, r_5, r_6\}$ and $\Delta r_6 < 0$. After removing r_6, we get the stable marriage $\mathcal{M}_{14}$ where $C(\mathcal{M}_{14}) < C(\mathcal{M}_{17})$. In effect, a matching derived from the k best non-negative antichains is likely to preserve good matching quality. We use the example rotation graph G_R to explain this observation. Table 7 shows the 4-best results in our example, where there are two candidates having the same cost. Remarkably, all antichains of the 4-best candidates are composed of non-negative rotations only, as $\{r_4, r_5, r_6, r_8\} \subset R^{\geq +}$. In the following, we analyze the cost difference between ENUM*-ϵ and ENUM*.

Theorem 3 (Error bound, ϵ). *The difference between the cost of the k-th matching returned by ENUM*-ϵ, $\mathcal{M}_\epsilon^k$, and the actual k-th matching, $\mathcal{M}^k$, is bounded by*

$$C(\mathcal{M}_\epsilon^k) - C(\mathcal{M}^k) \leq \Delta\mathcal{S}^1 - \Delta\mathcal{S}_\epsilon^k + \max \Delta r^- \tag{8}$$

where $\Delta\mathcal{S}_\epsilon^k = \sum_{r \in \mathcal{S}_\epsilon^k} \Delta r$ *and* $\max \Delta r^- = \max_{r^- \in R^-} \Delta r^-$.

Proof. Given the k-th matching returned by ENUM* and ENUM*-ϵ, we can calculate their cost gap by Equation 3.

$$C(\mathcal{M}_\epsilon^k) - C(\mathcal{M}^k) = \Delta\mathcal{S}^k - \Delta\mathcal{S}_\epsilon^k \tag{9}$$

From Equation 3, we also know that the top matching $\mathcal{M}^1$ has the largest $\Delta\mathcal{S}$ value, $\Delta\mathcal{S}^1$. To transform any matching (e.g., $\mathcal{M}^k$) to the top matching $\mathcal{M}^1$, we should detach at least one negative rotation therefrom, rendering the cost of its closed subset larger. $\Delta\mathcal{S}^1$ remains larger if we exclude from $\Delta\mathcal{S}^k$ the rotation of the *maximum* negative value, which has the smallest possible effect on $\Delta\mathcal{S}^k$:

$$\Delta\mathcal{S}^k - \max \Delta r^- \leq \Delta\mathcal{S}^1 \tag{10}$$

Combining Eq. 9 and Eq. 10, we get $C(\mathcal{M}_\epsilon^k) - C(\mathcal{M}^k) \leq \Delta\mathcal{S}^1 - \Delta\mathcal{S}_\epsilon^k + \max \Delta r^-$. □

We can easily compute this after generating the k-th result of ENUM*-ϵ. We derive $\Delta\mathcal{S}^1$ from the closed subset of the best matching, which is the first result returned by ENUM*-ϵ, and calculate $\max \Delta r^-$ in $O(n^2)$ time by means of a linear scan of the rotations.

ENUM*-LR-ϵ. Both the layer lower bound $\mathbf{LB}_\ell$ and the rotation lower bound $\mathbf{RB}_r$ work seamlessly with the heuristic enumeration, since both hold for the antichain set generated by the set of non-negative rotations $R^{\geq +} \subseteq R$. The error bound is identical to that of ENUM*-ϵ.

Stability and Complexity. ENUM*-ϵ is stable since it computes an antichain of non-negative rotations and then finds the corresponding closed subset in the original rotation graph G. In other words, ENUM*-ϵ provides a solution to k-ESMP in the subspace $R^{\geq +} \subseteq R$. Its time complexity is $O(n^2 + Q \cdot \mathcal{H}_\epsilon + \mathcal{U} \cdot \log k)$. The search space $\mathcal{H}_\epsilon$ is at most $\sum_{\alpha=1}^{n/2} \binom{|R^{\geq +}|}{\alpha}$. The exploration space of ENUM*-ϵ is significantly smaller than that of ENUM*, since $R^{\geq +}$ is a portion of the rotation set R. In our experiments, the size of $R^{\geq +}$ is about 50% of that of R, while $R^{\geq +}$ generates about 1%-10% of the antichains generated by R.

5 EXPERIMENTS

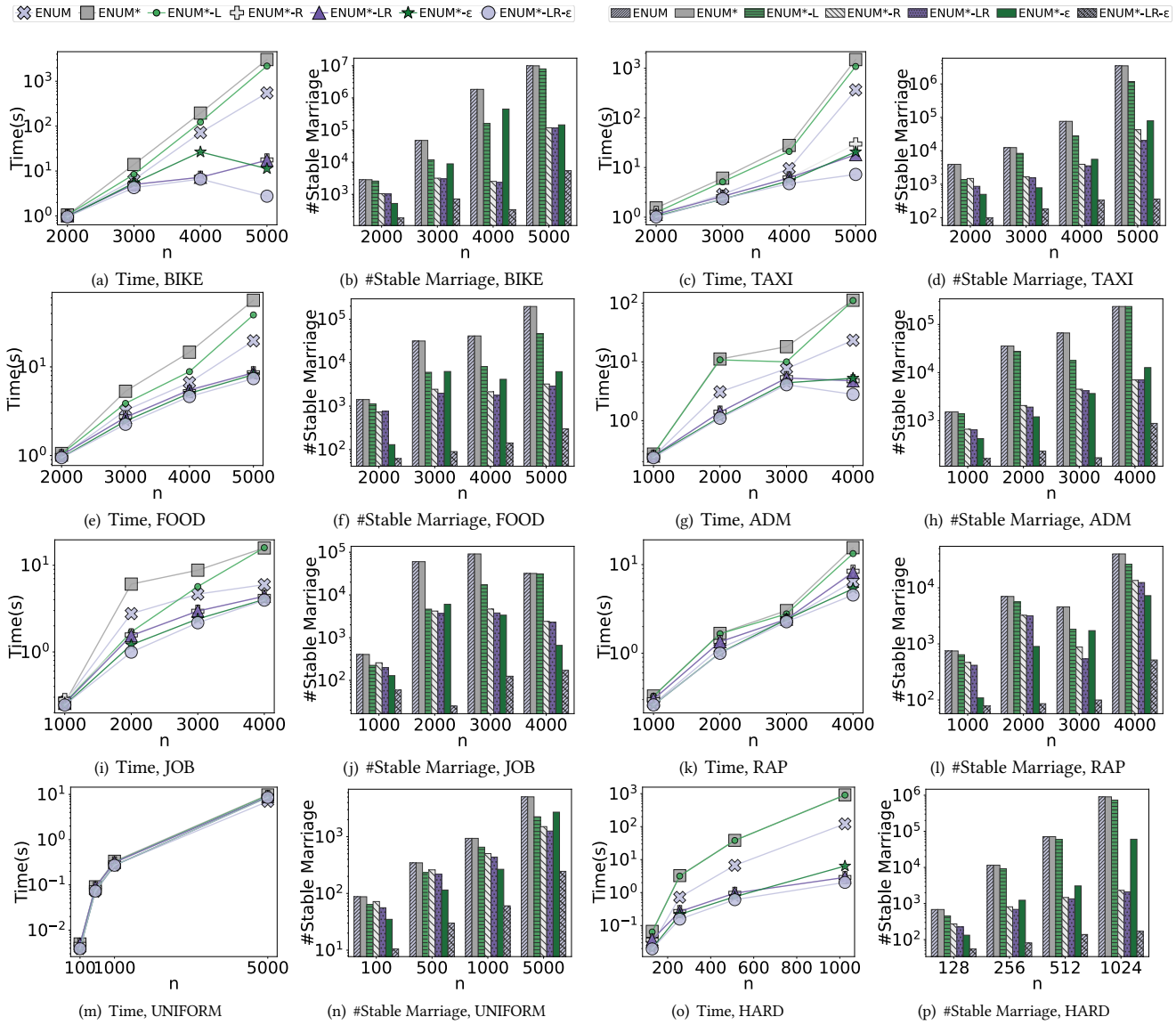
We conduct a thorough experimental study on our enumeration algorithm and bounding techniques.

5.1 Experimental Settings

Datasets. We use the following data. Table 8 lists the instance sizes and Table 9 lists statistics on G for a portion of data settings.

- **BIKE**. We use bike sharing data[2] to calculate distances from a start point to an end point as agent preferences on the X side. Agent preferences on the Y side are the values of orders for a bike on its start point; order values of follow a uniform distribution.
- **TAXI**/**TAXI+**. As with the **BIKE** dataset, we construct a two-sided market from taxi and user data in the NYC Taxi dataset[3].
- **FOOD**. From the restaurant data[4], we collect the locations of restaurants and customers to calculate the distance and extract the rating from restaurants.

[2] bikeshare.metro.net/about/data/
[3] www.nyc.gov/site/tlc/about/data.page
[4] kaggle.com/datasets/uciml/restaurant-data-with-consumer-ratings



(a) Time, BIKE (b) #Stable Marriage, BIKE (c) Time, TAXI (d) #Stable Marriage, TAXI

(e) Time, FOOD (f) #Stable Marriage, FOOD (g) Time, ADM (h) #Stable Marriage, ADM

(i) Time, JOB (j) #Stable Marriage, JOB (k) Time, RAP (l) #Stable Marriage, RAP

(m) Time, UNIFORM (n) #Stable Marriage, UNIFORM (o) Time, HARD (p) #Stable Marriage, HARD

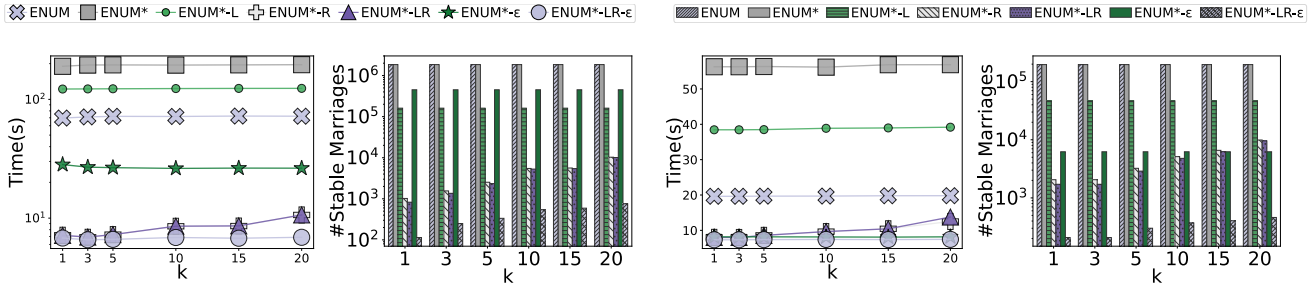
Figure 6: Effect of n ($k = 5$)

- **ADM**. University admission forms a classic scenario for the stable marriage problem [2, 43]. We obtain university ranking data[5] and GRE and TOEFL score from anonymous admission data[6]. We construct preference lists by a two-order sort, first by type of institute, then by rank within each type.
- **JOB**. Person-fit is the core task in online recruitment platforms [11, 56]. We construct a two-sided market using work experience[7] and salary[8] as the preferences of recruiters and job hunters.
- **RAP**. Reviewer-paper assignment is a common service in conference/journal systems. We derive preferences from affinity and CoI scores using topics and collaboration networks in Aminer [45][9].
- **UNIFORM**. The preference list follow a uniform distribution.
- **HARD**. An easy instance may contain few stable marriages if individuals have many common preferences [12, 39]. To demonstrate performance in hard instances (i.e., worse cases), we construct hard instances following previous studies [23, 44].

Evaluated Algorithms and Metrics. We implement all algorithms discussed in Section 4.

- **Mincut** [24]: an $O(n^4)$ network-flow algorithm that finds the best matching $\mathcal{M}^1$ (used for $k = 1$, see Section 3.3).
- **ENUM [19], ENUM***: brute-force enumeration algorithms.
- **ENUM*-L, ENUM*-R, ENUM*-LR**: improved algorithms with pruning techniques, i.e., applying **LB**, **RB** and **LB+RB** on ENUM*.
- **ENUM*-ϵ, ENUM*-LR-ϵ**: heuristic algorithms.

[5] kaggle.com/datasets/mylesoneill/world-university-rankings
[6] kaggle.com/datasets/akshaydattatraykhare/data-for-admission-in-the-university
[7] kaggle.com/datasets/arashnic/hr-analytics-job-change-of-data-scientists
[8] www.kaggle.com/datasets/andrewmvd/data-scientist-jobs
[9] aminer.cn/aminernetwork



(c) Time, BIKE (d) #Stable Marriage, BIKE (e) Time, FOOD (f) #Stable Marriage, FOOD

Figure 7: Effect of k (n = 5000)

Table 8: Dataset parameters (1 instance for each n in TAXI+, 10 instances for each n in others)

Dataset	n	X/Y	Preferences
BIKE	2000, 3000, 4000, 5000	bike/user	distance/utility
TAXI	2000, 3000, 4000, 5000	taxi/user	distance/utility
TAXI+	10000, 50000, 100000		
FOOD	2000, 3000, 4000, 5000	task/worker	distance/rating
ADM	1000, 2000, 3000, 4000	college/student	ranking/grades
JOB		job/person	salary/experi
RAP		paper/reviewer	affinity/COI
UNIFORM	100, 500, 1000, 5000	/	/
HARD	128, 256, 512, 1024	/	/

Table 9: Statistics of G_R

Dataset	n	$\|R\|$	$\|E\|$	$\|R^+\|$	$\|L\|$
BIKE	5000	426	9284	219	221
FOOD	4000	437	3205	245	251
ADM	2000	248	3355	117	148
HARD	1024	454	20686	248	250

Table 10: Taxi stat. of $\mathcal{U}$

$n, k = 5$	2000	3000	4000	5000
ENUM*	3.20%	1.27%	0.32%	<0.01%
ENUM*-R	3.20%	1.27%	0.27%	<0.01%
ENUM*-ϵ	1.53%	0.73%	0.16%	<0.01%

Implementation. We performed experiments on an Intel i9-9900K machine @3.60GHz, with 64G memory running Linux. All methods were implemented in C++. All evaluated methods, the data script, and the evaluated datasets can be found in our Github repo[10].

5.2 Efficiency

Varying n. We investigate the effect of instance size n with $k = 5$ on the datasets in Table 8. Figure 6 presents our results on the time of exact algorithms, the time of heuristic algorithms, and the number of enumerated stable marriages. The results reconfirm the time complexity analysis of ENUM and ENUM* in Section 4.3, by which ENUM performs better in brute-force enumeration. However, as Figures 6(a), 6(b) illustrate, our best exact method ENUM*-LR outperforms the non-pruning method ENUM by up to two orders of magnitude, as the layer bound **LB** (Section 4.1) and the rotation bound **RB** (Section 4.2) prune a large set of stable marriages; the same figures also show the efficiency of heuristics ENUM*-ϵ and ENUM*-LR-ϵ, which even improve their runtime as n grows when the corresponding number of stable marriages N falls. However, the non-pruning heuristic ENUM*-ϵ performs *worse* than the exact solution ENUM*-LR when the number of stable marriages N is large (i.e., $N \geq 10^5$). This result vindicates the significance of our bounding techniques. Likewise, ENUM*-LR-ϵ outperforms ENUM*-ϵ by 1–2 orders of magnitude. Still, as Figures 6(k), 6(l), 6(m) and 6(n) indicate, these bounding techniques are less effective in the RAP and UNIFORM datasets, which have fewer stable marriages. We record the counts of updates for our algorithms and calculate the percentage of updates vs. the number of stable marriages $\mathcal{U}/N$; the results in Table 10 justify the cost of updating the k-best results.

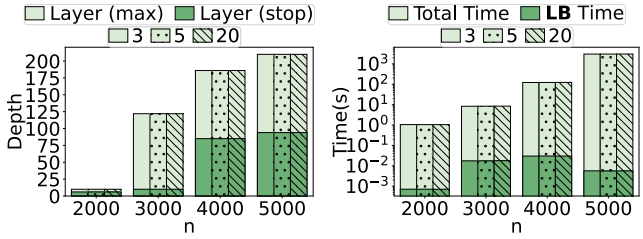
Varying k. We evaluate all proposed methods by varying k in {1, 3, 5, 10, 15, 20}. The Mincut algorithm takes 0.0134 seconds in BIKE and 0.0372 seconds in FOOD. While for $k = 1$, the ESMP can be solved in polynomial time (Section 3.3), for the sake of completeness, we also evaluate our algorithms with $k = 1$. Figure 7 shows our results. ENUM* and ENUM*-ϵ should be insensitive to k since the only overhead of k is to maintain the top-k rank results in a heap. The runtime of other methods, ENUM*-R, ENUM*-LR, and ENUM*-LR-ϵ, increases with the value of k. For instance, in the BIKE dataset, ENUM*-LR takes 6.92 seconds when $k = 5$ and increases to 10.64 seconds seconds when $k = 20$. This moderate growth demonstrates the robustness of our bounding techniques.

5.3 Pruning Techniques

Effect of layer bound $\mathbf{LB}_\ell$. Figure 8 shows how the layer bound performs on different problem instances. ENUM*-L (stop) explores about half the layers compared to ENUM (max). However, the complexity of stable marriage enumeration grows exponentially with the number of layers. Considering two instances, G_1 consisting of 32 layers and G_2 consisting of 338 layers, the enumeration on G_1 terminates at layer 10, while on G_2 it terminates at layer 169. As a result of layer bounds, the search space size on G_1 shrinks from 1,419 down to 1,125, while on G_2 it drops from 4,169,824 to 1,317,888. As Figure 7 shows, the response time of ENUM*-L is 10% faster than that of ENUM* on average; the cost of calculating **LB** is justified considering the total search time in Figure 8(b).

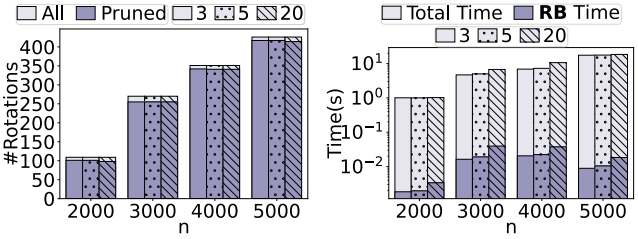
Effect of rotation bound $\mathbf{RB}_r$. Figure 9(a) shows the number of rotations pruned by the rotation lower bound. In the BIKE data, on average 97.18% of rotations are pruned due to **RB**, delivering a 1–2 orders of magnitude improvement in execution time. Notably, the

[10] github.com/Asuka54089/KESMP



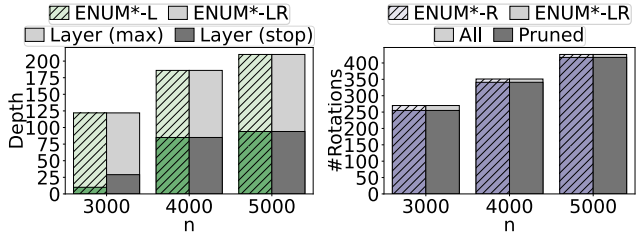
(a) Stop Layer, BIKE (b) Calculate **LB**, BIKE

Figure 8: Statistics on the layer bound LB



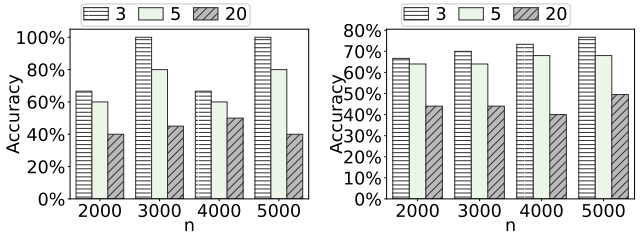
(a) #r (Pruned), BIKE (b) Calculate **RB**, BIKE

Figure 9: Statistics on the rotation bound RB



(a) Stop Layer, BIKE (b) #r (Pruned), BIKE

Figure 10: Statistics on LB+RB ($k = 5$)



(a) BIKE (b) FOOD

Figure 11: Accuracy

pruning ratio is higher than that due to the layer bound, since the number of rotations is much higher than the number of layers in G, hence the cost of calculating **RB** is also justified in Figure 9(b).

Effect of LB$_\ell$+RB$_r$. To study the effect of the two bounds, we show some statistics on the uniform datasets in Figure 10. We observe that ENUM*-LR outperforms other methods, with a trend similar to those seen in Figures 8 and 9.

Table 11: Comparison of $\mathcal{M}^i$ vs. the best matching $\mathcal{M}^1$

i	FOOD (n = 3000)			HARD (n = 1024)		
	$\#r$	$\#(x,y)$	ΔC	$\#r$	$\#(x,y)$	ΔC
2	1	6	14	1	4	1
3	1	25	72	1	4	11
5	2	45	332	1	2	270
10	4	65	460	2	14	314
20	9	119	803	5	61	471

5.4 Effectiveness

Statistics on exact matchings. To show the effectiveness of the k-best result, we compare the i-th matching result $\mathcal{M}^i$ to the best result, $\mathcal{M}^1$, where $i \in \{2, 3, 5, 10, 20\}$. Table 11 shows (1) the number of rotation changes $\#r$, (2) the number of pair changes $\#(x, y)$, and (3) the egalitarian cost difference $\Delta C = C(\mathcal{M}^k) - C(\mathcal{M}^1)$. Even though a matching may consist of more than 10^3 pairs, a decision maker may check the matchings at different positions in the ranking by inspecting only the pairs in the associated rotations. In the FOOD dataset, the number of pair changes (and the cost difference) is more sensitive to i than in other datasets. This result corroborates the difficulty of k-ESMP.

Statistics on heuristic matchings. We also compare the exact matching result $\mathcal{M}^i$ to the heuristic result $\mathcal{M}^i_\epsilon$ ($i \in \{1, 2, 3, 5, 10\}$). Table 12 shows the difference in terms of rotations $\#r$, pairs $\#(x, y)$, egalitarian cost gap $\Delta C_\epsilon = C(\mathcal{M}^i_\epsilon) - C(\mathcal{M}^i)$ and relative [29] egalitarian cost gap $\Delta C_\epsilon / C(\mathcal{M}^i)$. For small i, the gap between the heuristic and the exact solution is narrow.

Table 12: Comparison of $\mathcal{M}^i$ vs. heuristic matching $\mathcal{M}^i_\epsilon$

i	FOOD (n = 3000)				HARD (n = 1024)			
	$\#r$	$\#(x,y)$	ΔC_ϵ	$\Delta C_\epsilon / C(\mathcal{M}^i)$	$\#r$	$\#(x,y)$	ΔC_ϵ	$\Delta C_\epsilon / C(\mathcal{M}^i)$
1	0	0	0	0	0	0	0	0
2	2	9	123	0.0746%	2	8	10	0.0227%
3	2	15	228	0.1383%	2	6	43	0.0985%
5	20	362	1105	0.6706%	2	12	259	0.5897%
10	29	424	1109	0.6731%	3	10	280	0.6420%

Ranking Quality. Lastly, we compare the exact k-best matching list to the heuristic k-best matching list with accuracy and Normalized Discounted Cumulative Gain (NDCG@k) [28]; we define *accuracy* as follows:

$$Acc(\mathbb{M}_\epsilon(k), \mathbb{M}(k)) = \frac{1}{k} |\{C(\mathcal{M}^1_\epsilon), ..., C(\mathcal{M}^k_\epsilon)\} \cap \{C(\mathcal{M}^1), ..., C(\mathcal{M}^k)\}|$$

where two matching results are considered identical if they share the same egalitarian cost. Figure 11 shows our results. As k grows, the accuracy of the heuristic solution drops. Overall, our heuristic achieves ~40%-80% accuracy while its response time is 1–2 orders of magnitude faster as observed in Figures 6 and 7.

We also compare the *normalized discounted cumulative gain* (NDCG@k) between $\mathbb{M}(k)$ and $\mathbb{M}_\epsilon(k)$ ($k \in \{5, 20\}$) to assess the quality of the heuristic. We define a relevance score in $[0, 1]$, *Egalitarian Contentment*, by normalizing the egalitarian cost, as $EC(\mathcal{M}) = 1 - \frac{C(\mathcal{M})}{2n^2}$. NDCG is then defined as:

$$NDCG(\mathbb{M}_\epsilon(k), \mathbb{M}(k)) = \frac{DCG}{IDCG}, \tag{11}$$

where $DCG = \sum_i^k \frac{2^{EC(\mathcal{M}^i_\epsilon)} - 1}{\log_2(i+1)}$ and $IDCG = \sum_i^k \frac{2^{EC(\mathcal{M}^i)} - 1}{\log_2(i+1)}$.

Table 13 shows our NDCG@k results (on **FOOD** and **HARD** datasets only due to space limit), which are all higher than 0.99, even while the heuristic solutions ENUM*-ϵ and ENUM*-LR-ϵ are more efficient than the exact solution. We conclude that our heuristics achieve a good balance between efficiency and effectiveness.

Table 13: NDCG results

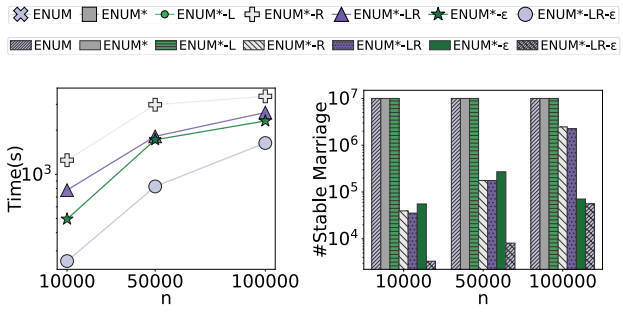
N	Dataset:FOOD		N	Dataset:HARD	
	NDCG@5	NDCG@20		NDCG@5	NDCG@20
2000	0.9999	0.9999	128	0.9991	0.9998
3000	0.9999	0.9999	256	0.9999	0.9999
4000	0.9999	0.9999	512	0.9999	0.9990
5000	0.9999	0.9998	1024	0.9998	0.9992

5.5 Scalability

To the best of our knowledge, recent works on task assignment and stable matchings that considers asymmetric two-sided preference typically set the number of agents (i.e., the instance size n) in the order of magnitude of 10^3 [17, 31, 50, 60, 61]. The storage of preference lists on rotations poses a significant memory challenge. For example, an instance with $n = 10^5$ requires approximately $2 \times 4 \times 10^{10}$ bytes, which is equivalent to approximately 74GB.

To evaluate our methods on large-scale datasets, we implemented a progressive disk-based loading mechanism for preference lists during rotation graph construction. Specifically, we store the preference lists on disk and load each preference list on demand, when needed. The time complexity of this loading process is $O(n^2)$. As Table 9 indicates, the size of the rotation graph, represented by the number of rotations $|R|$, is typically much smaller than the theoretically expected $O(n^2)$ (Table 4) in practice.

Figure 12 presents the scalability of our methods on **TAXI+** data. Since counting all stable marriages becomes intractable as N grows, we terminate algorithms ENUM, ENUM*, and ENUM*-L, when their running time goes up to one hour, which indicates all these instances have at least 10^7 stable marriages. Still, even as n grows to 10^5, two exact algorithms, ENUM*-R and ENUM*-LR, and two heuristics, ENUM*-ϵ and ENUM*-LR-ϵ, remain operational.



(c) Time, TAXI+ (d) #Stable Marriage, TAXI+

Figure 12: Scalability ($k = 5$)

5.6 A Use Case for k-best Assignments

Now we study a stable marriage instance with $n = 3000$ in dataset **BIKE** to discuss (i) the necessity of finding k-ESMP and (ii) how to use the k-best assignments in real applications.

Figure 13 shows the 3-best task assignments and their egalitarian costs $C(\mathcal{M}^1) < C(\mathcal{M}^2) < C(\mathcal{M}^3)$. We report the rotations in the k-best stable marriages and their corresponding closed subsets to facilitate decision-making. For instance, the transformation from $\mathcal{M}^1$ to $\mathcal{M}^2$ re-assigns pairs $(1790, 586), (2931, 503)$ to $(1790, 503), (2931, 586)$. Since other assigned pairs remain unchanged in $\mathcal{M}^1$ and $\mathcal{M}^2$, a decision maker may pick a result by only checking the changed pairs in the rotation structure. In other task assignment problems, the re-assigned pairs lack this inner connection. Furthermore, both $\mathcal{M}^2$ and $\mathcal{M}^3$ have a higher egalitarian cost than $\mathcal{M}^1$, with a lower difference of satisfaction (i.e., the sum of the positions for choices) between the task side and the worker side. Our k-ESMP gives decision makers a collection of high-quality findings to prevent them from overlooking practical good tasks.

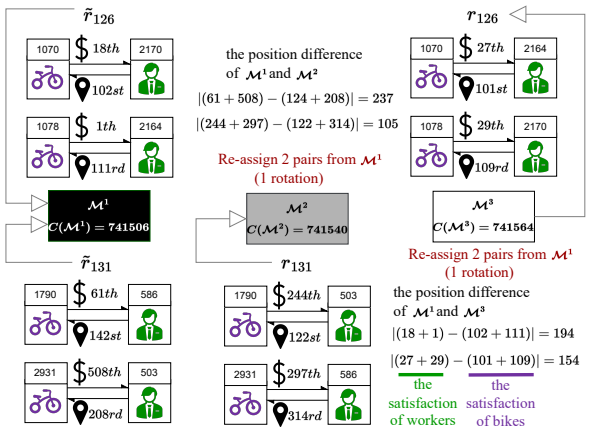


Figure 13: A Use Case for 3-best Assignments.

6 CONCLUSION

We proposed and studied the **NP**-hard k-best egalitarian stable marriage problem for task assignment, which seeks a set of high-quality matchings among workers and tasks based on egalitarian cost. To our knowledge, this is the first work that extends the egalitarian stable marriage problem to a k-best formulation. To efficiently find matchings of high cumulative satisfaction, we proposed several enhancements upon a brute-force enumeration harnessing the topological structure of rotations and two effective lower-bounding techniques, layer lower bound and rotation lower bound. Further, we designed a heuristic yet stable solution. Our experimental study demonstrates the efficiency and effectiveness of our solutions on spatial and non-spatial datasets. In the future, we will study fair spatial matchings on other variants of stable marriage problems, such as *stable roommates*.

ACKNOWLEDGMENTS

This work was supported by the Science and Technology Development Fund of Macau SAR (0015/2019/AKP, 0031/2022/A, SKL-IOTSC-2021-2023), the RG MYRG2022-00252-FST from the University of Macau, and Wuyi University Hong Kong and Macau joint Research Fund 2021WGALH14. We also thank the Danish Agency for Higher Education and Science for their support (Ref. no.: 1113-00030B).

REFERENCES

[1] Georgios Askalidis, Nicole Immorlica, Augustine Kwanashie, David F. Manlove, and Emmanouil Pountourakis. 2013. Socially Stable Matchings in the Hospitals/Residents Problem. In *Algorithms and Data Structures - 13th Intl Symposium (WADS)*. 85–96.

[2] Mourad Baïou and Michel Balinski. 2004. Student admissions and faculty recruitment. *Theor. Comput. Sci.* 322, 2 (2004), 245–265.

[3] Péter Biró, Avinatan Hassidim, Assaf Romm, Ran I. Shorrer, and Sándor Sovago. 2022. The Large Core of College Admission Markets: Theory and Evidence. In *EC '22: The 23rd ACM Conference on Economics and Computation*. ACM, 958–959.

[4] Chandra R. Chegireddy and Horst W. Hamacher. 1987. Algorithms for finding k-best perfect matchings. *Discret. Appl. Math.* 18, 2 (1987), 155–165.

[5] Zhao Chen, Peng Cheng, Lei Chen, Xuemin Lin, and Cyrus Shahabi. 2020. Fair Task Assignment in Spatial Crowdsourcing. *Proc. VLDB Endow.* 13, 11 (2020), 2479–2492.

[6] Peng Cheng, Lei Chen, and Jieping Ye. 2019. Cooperation-Aware Task Assignment in Spatial Crowdsourcing. In *ICDE*. 1442–1453.

[7] David Eppstein. 1998. Finding the k Shortest Paths. *SIAM J. Comput.* 28, 2 (1998), 652–673.

[8] David Eppstein. 2016. k-Best Enumeration. In *Encycl. of Alg.* 1003–1006.

[9] Patricia Everaere, Maxime Morge, and Gauthier Picard. 2013. Minimal concession strategy for reaching fair, optimal and stable marriages. In *AAMAS*. 1319–1320.

[10] Ralph Freese, Jaroslav Ježek, and James Bryant Nation. 1995. *Free lattices*. Vol. 42. American Mathematical Soc.

[11] Bin Fu, Hongzhi Liu, Yao Zhu, Yang Song, Tao Zhang, and Zhonghai Wu. 2021. Beyond Matching: Modeling Two-Sided Multi-Behavioral Sequences for Dynamic Person-Job Fit. In *DASFAA*. 359–375.

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

[13] David Gale and Marilda Sotomayor. 1985. Some remarks on the stable matching problem. *Discret. Appl. Math.* 11, 3 (1985), 223–232.

[14] Mirco Gelain, Maria Silvia Pini, Francesca Rossi, Kristen Brent Venable, and Toby Walsh. 2010. Local Search for Stable Marriage Problems with Ties and Incomplete Lists. In *PRICAI 2010: Trends in Artificial Intelligence, 11th Pacific Rim International Conference on Artificial Intelligence*, Vol. 6230. 64–75.

[15] Begum Genc, Mohamed Siala, Barry O'Sullivan, and Gilles Simonin. 2017. Finding Robust Solutions to Stable Marriage. In *36th International Joint Conference on Artificial Intelligence (IJCAI)*. 631–637.

[16] Begum Genc, Mohamed Siala, Barry O'Sullivan, and Gilles Simonin. 2017. Robust Stable Marriage. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*. 4925–4926.

[17] Ioannis Giannakopoulos, Panagiotis Karras, Dimitrios Tsoumakos, Katerina Doka, and Nectarios Koziris. 2015. An Equitable Solution to the Stable Marriage Problem. In *ICTAI*. 989–996.

[18] Dan Gusfield. 1987. Three Fast Algorithms for Four Problems in Stable Marriage. *SIAM J. Comput.* 16, 1 (1987), 111–128.

[19] Dan Gusfield and Robert W. Irving. 1989. *The Stable marriage problem - structure and algorithms*. MIT Press.

[20] Avinatan Hassidim, Assaf Romm, and Ran I. Shorrer. 2017. Need vs. Merit: The Large Core of College Admissions Markets. Available at SSRN: https://ssrn.com/abstract=3071873.

[21] Robert W. Irving. 1985. An Efficient Algorithm for the "Stable Roommates" Problem. *J. Algorithms* 6, 4 (1985), 577–595.

[22] Robert W. Irving. 1994. Stable Marriage and Indifference. *Discret. Appl. Math.* 48, 3 (1994), 261–272.

[23] Robert W. Irving and Paul Leather. 1986. The Complexity of Counting Stable Marriages. *SIAM J. Comput.* 15, 3 (1986), 655–667.

[24] Robert W. Irving, Paul Leather, and Dan Gusfield. 1987. An efficient algorithm for the "optimal" stable marriage. *J. ACM* 34, 3 (1987), 532–543.

[25] Kazuo Iwama, David F. Manlove, Shuichi Miyazaki, and Yasufumi Morita. 1999. Stable Marriage with Incomplete Lists and Ties. In *Automata, Languages and Programming, 26th International Colloquium, (ICALP)*. 443–452.

[26] Kazuo Iwama and Shuichi Miyazaki. 2008. A survey of the stable marriage problem and its variants. In *International conference on informatics education and research for knowledge-circulating society (ICKS 2008)*. IEEE, 131–136.

[27] Kazuo Iwama, Shuichi Miyazaki, and Hiroki Yanagisawa. 2010. Approximation algorithms for the sex-equal stable marriage problem. *ACM Trans. Algorithms* 7, 1 (2010), 2:1–2:17.

[28] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.* 20, 4 (2002), 422–446.

[29] Panagiotis Karras and Nikos Mamoulis. 2008. Hierarchical synopses with optimal error guarantees. *ACM Trans. Database Syst.* 33, 3 (2008), 18:1–18:53.

[30] Akiko Kato. 1993. Complexity of the sex-equal stable marriage problem. *Japan Journal of Industrial and Applied Mathematics* 10, 1 (1993), 1–19.

[31] Ramneek Kaur, Vikram Goyal, Venkata M. V. Gunturi, and Cheng Long. 2022. A Matching Based Spatial Crowdsourcing Framework for Egalitarian Task Assignment. In *IEEE MDM*. 185–187.

[32] Leyla Kazemi and Cyrus Shahabi. 2012. GeoCrowd: enabling query answering with spatial crowdsourcing. In *SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 189–198.

[33] Donald Ervin Knuth. 1976. Marriages stables. *Technical report* (1976).

[34] Donald Ervin Knuth. 1997. *Stable marriage and its relation to other combinatorial problems : an introduction to the mathematical analysis of algorithms*. American Mathematical Society.

[35] Ngai Meng Kou, Leong Hou U, Nikos Mamoulis, and Zhiguo Gong. 2015. Weighted Coverage based Reviewer Assignment. In *SIGMOD*. 2031–2046.

[36] Cheng Long, Raymond Chi-Wing Wong, Yu Peng, and Liangliang Ye. 2013. On Good and Fair Paper-Reviewer Assignment. In *IEEE ICDM*. 1145–1150.

[37] Varun S. Malhotra. 2004. On the Stability of Multiple Partner Stable Marriages with Ties. In *Algorithms - ESA 2004, 12th Annual European Symposium*. 508–519.

[38] Eric McDermid and Robert W. Irving. 2014. Sex-Equal Stable Matchings: Complexity and Exact Algorithms. *Algorithmica* 68, 3 (2014), 545–570.

[39] D. G. McVitie and L. B. Wilson. 1971. The Stable Marriage Problem. *Commun. ACM* 14, 7 (1971), 486–490.

[40] Katta G. Murty. 1968. An Algorithm for Ranking all the Assignments in Order of Increasing Cost. *Oper. Res.* 16, 3 (1968), 682–687.

[41] Dimitris Papadias, Yufei Tao, Kyriakos Mouratidis, and Chun Kit Hui. 2005. Aggregate nearest neighbor queries in spatial databases. *ACM Trans. Database Syst.* 30, 2 (2005), 529–576.

[42] J. Scott Provan and Michael O. Ball. 1983. The Complexity of Counting Cuts and of Computing the Probability that a Graph is Connected. *SIAM J. Comput.* 12, 4 (1983), 777–788.

[43] A. F. M. Saifuddin Saif, Mokaddesh Rashid, Imran Ziahad Bhuiyan, Md. Wasim Sajjad Ifty, and Md. Rawnak Sarker. 2020. Stable Marriage Algorithm for Student-College Matching with Quota Constraints. In *ICCA*. 52:1–52:5.

[44] Mohamed Siala and Barry O'Sullivan. 2017. Rotation-Based Formulation for Stable Matching. In *Principles and Practice of Constraint Programming - 23rd International Conference (CP)*. 262–277.

[45] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. ArnetMiner: extraction and mining of academic social networks. In *KDD*. 990–998.

[46] Chung-Piaw Teo and Jay Sethuraman. 1998. The Geometry of Fractional Stable Matchings and Its Applications. *Math. Oper. Res.* 23, 4 (1998), 874–891.

[47] Yongxin Tong, Jieying She, Bolin Ding, Lei Chen, Tianyu Wo, and Ke Xu. 2016. Online Minimum Matching in Real-Time Spatial Data: Experiments and Analysis. *Proc. VLDB Endow.* 9, 12 (2016), 1053–1064.

[48] Yongxin Tong, Zimu Zhou, Yuxiang Zeng, Lei Chen, and Cyrus Shahabi. 2020. Spatial crowdsourcing: a survey. *VLDB J.* 29, 1 (2020), 217–250.

[49] Nikolaos Tziavelis, Ioannis Giannakopoulos, Katerina Doka, Nectarios Koziris, and Panagiotis Karras. 2019. Equitable Stable Matchings in Quadratic Time. In *NeurIPS*. 455–465.

[50] Nikolaos Tziavelis, Ioannis Giannakopoulos, Rune Quist Johansen, Katerina Doka, Nectarios Koziris, and Panagiotis Karras. 2020. Fair Procedures for Fair Stable Marriage Outcomes. In *AAAI*. 7269–7276.

[51] Leong Hou U, Nikos Mamoulis, and Man Lung Yiu. 2008. Computation and Monitoring of Exclusive Closest Pairs. *IEEE TKDE* 20, 12 (2008), 1641–1654.

[52] Leong Hou U, Kyriakos Mouratidis, and Nikos Mamoulis. 2010. Continuous spatial assignment of moving users. *VLDB J.* 19, 2 (2010), 141–160.

[53] Leong Hou U, Kyriakos Mouratidis, Man Lung Yiu, and Nikos Mamoulis. 2010. Optimal matching between spatial datasets under capacity constraints. *ACM Trans. Database Syst.* 35, 2 (2010), 9:1–9:44.

[54] Hoang Huu Viet, Le Hong Trang, SeungGwan Lee, and TaeChoong Chung. 2016. A Bidirectional Local Search for the Stable Marriage Problem. In *IEEE Intl Conference on Advanced Computing and Applications (ACOMP)*. 18–24.

[55] Hoang Huu Viet, Le Hong Trang, SeungGwan Lee, and TaeChoong Chung. 2016. An Empirical Local Search for the Stable Marriage Problem. In *PRICAI 2016*. 556–564.

[56] Chen Yang, Yupeng Hou, Yang Song, Tao Zhang, Ji-Rong Wen, and Wayne Xin Zhao. 2022. Modeling Two-Way Selection Preference for Person-Job Fit. In *RecSys*. 102–112.

[57] Ziqiang Yu, Xiaohui Yu, Nick Koudas, Yang Liu, Yifan Li, Yueting Chen, and Dingyu Yang. 2020. Distributed Processing of k Shortest Path Queries over Dynamic Road Networks. In *SIGMOD*. 665–679.

[58] Boming Zhao, Pan Xu, Yexuan Shi, Yongxin Tong, Zimu Zhou, and Yuxiang Zeng. 2019. Preference-Aware Task Assignment in On-Demand Taxi Dispatching: An Online Stable Matching Approach. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI*. 2245–2252.

[59] Yan Zhao, Kai Zheng, Jiannan Guo, Bin Yang, Torben Bach Pedersen, and Christian S. Jensen. 2021. Fairness-aware Task Assignment in Spatial Crowdsourcing: Game-Theoretic Approaches. In *ICDE*. 265–276.

[60] Yan Zhao, Kai Zheng, Hongzhi Yin, Guanfeng Liu, Junhua Fang, and Xiaofang Zhou. 2022. Preference-Aware Task Assignment in Spatial Crowdsourcing: From Individuals to Groups. *IEEE Trans. Knowl. Data Eng.* 34, 7 (2022), 3461–3477.

[61] Xu Zhou, Shiting Liang, Kenli Li, Yunjun Gao, and Keqin Li. 2022. Bilateral Preference-aware Task Assignment in Spatial Crowdsourcing. In *ICDE*. 1687–1699.