An Approach to Robustness in the Stable Roommates Problem and its Comparison with the Stable Marriage Problem

Begum Genc, Mohamed Siala, Gilles Simonin, Barry O'Sullivan

An Approach to Robustness in the Stable Roommates Problem and its Comparison with the Stable Marriage Problem

Begum Genc[1], Mohamed Siala[2], Gilles Simonin[3], and Barry O'Sullivan[1]

[1] Insight, Centre for Data Analytics, University College Cork, Cork Ireland
{begum.genc,barry.osullivan}@insight-centre.org
[2] LAAS-CNRS, Université de Toulouse, CNRS, INSA, Toulouse, France
mohamed.siala@laas.fr
[3] IMT Atlantique, LS2N, CNRS, UBL, 44307, Nantes, France
gilles.simonin@imt-atlantique.fr

Abstract. Recently a robustness notion for matching problems based on the concept of a (a, b)-supermatch is proposed for the Stable Marriage problem (SM). In this paper we extend this notion to another matching problem, namely the Stable Roommates problem (SR). We define a polynomial-time procedure based on the concept of reduced rotation poset to verify if a stable matching is a $(1, b)$-supermatch. Then, we adapt a local search and a genetic local search procedure to find the $(1, b)$-supermatch that minimises b in a given SR instance. Finally, we compare the two models and also create different SM and SR instances to present empirical results on the robustness of these instances.

1 Introduction

Robustness to change is an important property that has a variety of definitions in different settings [15]. There exist many robustness notions within the context of matching problems. These robustness notions mostly focus on handling uncertainty and erroneous data in the input [3,1,2,12]. Genc et al. introduced a novel notion of robustness for the Stable Marriage problem (SM) where the robustness of a solution refers to its capability to be repaired at a small bounded cost in case of an unforeseen event [4]. The notion of (a, b)-supermatches differs from the other robustness notions in this context since it specifies a degree of repairability. This property is often referred as fault-tolerance. The (a, b)-supermatch concept defines the notion of robustness for matching problems by using the fault-tolerance framework [8,9].

The SM is defined by a set of men and a set of women, each of which has a set of preferences over people of opposite sex. The task is to find a (monogamous) matching between men and women that is stable. A matching is said to be stable if there are no two pairs that are not matched to each other, but they prefer being together than being with their current partners. The robust variant of the problem is called *Robust Stable Marriage (RSM)* [4], in which the robustness of a stable matching is measured by the minimum number of changes required to obtain another stable matching in the case of break-up of some pairs. If a pair appears in all the stable matchings, the pair is said to be

fixed, otherwise, non-fixed. An (a, b)-supermatch is a stable matching such that if any a non-fixed agents (men/women) break-up, it is possible to find another stable matching by changing the partners of those a agents and also changing the partners of at most b others. The previous work on the RSM includes the proposal of the problem, a complexity study, a polynomial-time verification procedure for a given $(1, b)$-supermatch, and three different models (constraint programming, genetic algorithm, local search) to find the $(1, b)$-supermatch that minimises b for a given SM instance [6,4]. We investigate in this paper this robustness concept further on a generalised version of the SM, namely the Stable Roommates problem (SR). The Stable Roommates problem is a one-sided generalisation of SM, where any two agents regardless of their gender can be matched. We define the *Robust Stable Roommates problem (RSR)* analogous to the RSM. To the best of our knowledge, there is no previous research on finding the (a, b)-supermatches of the SR.

The motivation behind studying RSR is due to the large applicability of SR and the importance to handle the dynamism of the real world. Take the example of P2P networks where peers (computers for instance) are connected to each other for file sharing purposes [14]. Each peer has a preference list towards the other peers and a matching that respect stability is required. However, as the network evolves during time, peers continuously seek new partners. That is, if a peer that provides the file loses the connection, an alternative peer is needed for downloading a file. In this situation, we have have to maintain stability with possibly the minimum changes to the current solution. An (a, b)-supermatch guarantees finding other peers to the broken ones at a small number of additional changes while preserving stability.

The paper is organised as follows: In Section 2, we give a formal background and introduce the robust stable roommates problem. Then, in Section 3, we show that one can verify in polynomial time if a given stable matching (in the SR context) is a $(1, b)$-supermatch. Next, we adapt a local search procedure and a hybrid (genetic local search) model for finding robust solutions in Section 4. Finally, we present in Section 5 our empirical study.

2 Background and Notation

The Stable Roommates problem (SR) consists of a set of $2 \times n$ agents, where each agent has a preference list in which he/she ranks all other agents in strict order of preference. In this context, given a set of people P, a *matching* corresponds to a partition of P into disjoint pairs (or partners). A matching is *stable* if it admits no blocking pairs. A pair $\{p_i, p_j\}$ blocks a matching if: p_i is unassigned or prefers p_j to his/her current partner, or p_j is unassigned or prefers p_i to his/her current partner. The solution to an SR instance is a stable matching. If such a solution does not exist, then the instance is *unsolvable*. A pair is *stable* if it appears in some stable matching. If a pair appears in all stable matchings, it is called a *fixed* pair. If a person p has at least two different partners among all stable matchings, p is said to be *non-fixed*. We measure the distance between any two stable matchings M, M' by the number of different pairs $d(M, M') = \mid M \setminus M \mid$. The stable matching M' among all the stable matchings of the instance that has the minimum distance to M is said to be the *closest stable matching* to M.

Irving defines an $O(n^2)$ procedure to find a solution to SR or to report if none exists [10]. The procedure consists of two phases. Let us first define some notations to describe these phases. A *preference table* (denoted by T) is, for a given problem instance, a set of preference lists for which zero or more entries have been deleted. We use T_{init} to denote the initial preference table. During the two phases, some pairs are removed from T_{init}. We denote the preference list of a person p_i in a table T by $L_T(i)$. Let $f_T(p_i), s_T(p_i), l_T(p_i)$ denote the first, second and last entries of $L_T(i)$. The first phase is based on each person proposing to the first available person in their lists starting from T_{init} until every person has made a proposal that has been accepted, i.e. became semi-engaged. If a person p_i becomes semi-engaged to p_j, all pairs $\{p_j, p_k\}$ such that p_j prefers p_i to p_k are deleted from the table. The table obtained after applying the Phase 1 algorithm is called the *Phase-1 table* and is denoted by T_0.

The second phase of the algorithm is based on finding and eliminating *rotations* starting from T_0. A rotation ρ is a circular list denoted as $\rho = (x_0, y_0), (x_1, y_1), \ldots, (x_{r-1}, y_{r-1})$, where all $x_i, y_j \in P$. Each rotation has the property that $y_i = f_T(x_i)$ and $y_{i+1} = s_T(x_i)$ in a table T for all i, $0 \leq i \leq r-1$, where i+1 is taken modulo r. The set of people $\{x_0, \ldots, x_{r-1}\}$ is called the *X-set* of ρ, denoted by $X(\rho)$. Similarly, $\{y_0, \ldots, y_{r-1}\}$ is called the *Y-set* of ρ, denoted by $Y(\rho)$. Additionally, given a set of rotations R, $X(R) = \cup_{\rho \in R} X(\rho)$. Similar for the Y-set. The *elimination* of a rotation ρ from a table T means for each pair $\{p_i, p_j\}$, where $p_i = x_m$ and $p_j = y_m$ and $(x_m, y_m) \in \rho$, the deletion of $\{p_i, p_j\}$ and all pairs $\{y_m, z\}$ such that y_m prefers x_{m-1} to z from T. In this case, ρ is said to be *exposed* on T and the table after eliminating ρ is denoted by T/ρ. If after Phase 1 or Phase 2, all lists in T contain exactly one entry, then T represents a stable matching. Note that sometimes we use (p_i, p_j) and (x_m, y_m) interchangeably. The notation (x_m, y_m) is used for denoting the position of the pair (p_i, p_j) in ρ. Lemma 4.4.1 from [7] states that $\{p_i, p_j\}$ is a stable non-fixed pair if and only if (p_i, p_j) or (p_j, p_i) is in a non-singular rotation.

There are two types of rotations: *singular* and *non-singular*. A rotation $\rho = (x_0, y_0)$, $(x_1, y_1), \ldots, (x_{r-1}, y_{r-1})$ is called a non-singular rotation if $\bar{\rho} = (y_1, x_0), (y_2, x_1), \ldots$, (y_0, x_{r-1}) is also a rotation. In this case, ρ and $\bar{\rho}$ are called as *duals* of each other. If a rotation does not have a dual, then it is a singular rotation. We denote by T_S the table where all singular rotations are eliminated from T_0. A rotation ρ' is said to precede another rotation ρ (denoted by $\rho' \prec \rho$) if ρ' is eliminated for ρ to become exposed. In this case, we say ρ' is a *predecessor* of ρ and ρ is a *successor* of ρ'. A rotation ρ' is an *immediate predecessor* of ρ, and ρ is an *immediate successor* of ρ', if $\rho' \prec \rho$ and there does not exist a ρ^* such that $\rho' \prec \rho^* \prec \rho$. All predecessors and successors of a rotation, not necessarily immediate, are denoted by $N^-(\rho)$ and $N^+(\rho)$. The set of both singular and non-singular rotations under $\prec$ defines the *roommates rotation poset*. The set of non-singular rotations under $\prec$ defines the *reduced rotation poset* and is denoted by $\Pi = (\mathcal{V}, E)$. We refer to any two rotations as *incomparable* if none of them precede the other one, *comparable* otherwise. Let us illustrate these concepts on an SR instance $\mathcal{I}$. We use a sample instance of 10 people from page 180 in [7]. Fig. 1 represents the T_S of $\mathcal{I}$. Fig. 2 represents the reduced rotation poset of $\mathcal{I}$, where the pairs involved in the rotations are given next to their corresponding rotations for convenience.

Fig. 1. The T_S for an SR instance $\mathcal{I}$ of size 10.

p_i	$L_{T_S}(i)$
1	3 4 7
2	4 3 8 9
3	5 6 2 1
4	9 1 6 2
5	7 10 8 3
6	8 3 4 10
7	1 5
8	10 2 5 6
9	2 10 4
10	6 5 9 8

Fig. 2. Reduced rotation poset of $\mathcal{I}$ given in Table 1.

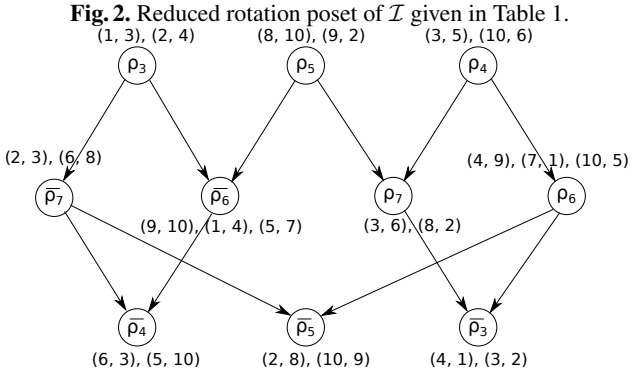

Table 1. A list of all the stable matchings and their corresponding complete closed subsets of $\mathcal{I}$.

$M_1 = \{(1,3),(2,4),(5,7),(6,8),(9,10)\}$	$S_1 = \{\bar{\rho_3}, \rho_4, \rho_5, \rho_6, \rho_7\}$
$M_2 = \{(1,7),(2,8),(3,5),(4,9),(6,10)\}$	$S_2 = \{\rho_3, \bar{\rho_4}, \rho_5, \bar{\rho_6}, \bar{\rho_7}\}$
$M_3 = \{(1,4),(2,9),(3,6),(5,7),(8,10)\}$	$S_3 = \{\rho_3, \rho_4, \bar{\rho_5}, \rho_6, \bar{\rho_7}\}$
$M_4 = \{(1,4),(2,3),(5,7),(6,8),(9,10)\}$	$S_4 = \{\rho_3, \rho_4, \rho_5, \rho_6, \rho_7\}$
$M_5 = \{(1,4),(2,8),(3,6),(5,7),(9,10)\}$	$S_5 = \{\rho_3, \rho_4, \rho_5, \rho_6, \bar{\rho_7}\}$
$M_6 = \{(1,7),(2,3),(4,9),(5,10),(6,8)\}$	$S_6 = \{\rho_3, \rho_4, \rho_5, \bar{\rho_6}, \rho_7\}$
$M_7 = \{(1,7),(2,8),(3,6),(4,9),(5,10)\}$	$S_7 = \{\rho_3, \rho_4, \rho_5, \bar{\rho_6}, \bar{\rho_7}\}$

A subset of the rotations in Π, containing one of each dual rotations and all their predecessors, is called a *complete closed subset*, denoted by S. There exists a 1-1 correspondence between the complete closed subsets of Π and the stable matchings of the underlying instance [7]. Any stable matching can be obtained by eliminating one of each dual rotations starting from T_S. A rotation ρ is said to *eliminate* $\{p_i, p_j\}$ if there exists a table T such that $\{p_i, p_j\} \in T$ and $\{p_i, p_j\} \notin T/\rho$. On the other hand, a rotation ρ is said to *produce* $\{p_i, p_j\}$ if there exists a table T such that $|L_T(i)| > 1$, $|L_T(j)| > 1$, $L_{T/\rho}(i)$ contains only p_j, and $L_{T/\rho}(j)$ contains only p_i. We use the term *flipping* ρ from S as the process of removing $\rho \in S$ from S and adding its dual $\bar{\rho}$ to S. A neighbour rotation ρ is $\rho \notin S$ and either the rotation has no predecessors ($N^-(\rho) = \emptyset$) or for all predecessors $\rho' \in N^-(S), \rho' \in S$. The set **N(*S*)** denotes the set of *neighbour rotations*. The set of all sink nodes of the graph induced by S is referred as the *sink rotations* of S, denoted as **L(*S*)**. Table 1 presents all the 7 stable matchings of $\mathcal{I}$ given in Fig. 1 and their corresponding complete closed subsets.

Throughout the paper, we denote by M a given stable matching, and its corresponding complete closed subset by S. If there are any subscripts or superscripts for M such as M_i^*, then they are applied to the corresponding complete closed subset (i.e. S_i^*). Lemma 1 and 2 are included here to be used in our proofs later.

Lemma 1 (Lemma 4.1.1 [7]). *Given an instance of the stable marriage problem involving n men and n women, there is an instance (in fact there are many instances) of the stable roommates problem involving those $2n$ persons such that the stable roommates matchings are precisely the stable matchings for the original SM instance.*

Lemma 2 (Lemma 4.3.7 [7]). *If ρ, σ are non-singular and π is a singular rotation, then: (1) $\rho \not\prec \bar{\rho}$; (2) $\rho \prec \sigma \iff \bar{\sigma} \prec \bar{\rho}$; (3) $\tau \prec \pi \implies \tau$ is singular.*

Robust Stable Roommates: We refer the problem of finding an (a, b)-supermatch to a given SR instance as the *Robust Stable Roommates problem (RSR)*. A stable matching of an RSR instance is called an (a, b)-*supermatch* if any a non-fixed pairs do not want to be partners anymore (i.e. *leave* the stable matching), it is possible to find another stable matching by changing the partners of the people involved in those a pairs and at most b other pairs.

Definition 1 ((a, b)-supermatch). *Given an SR instance $\mathcal{I}$, and two positive integers $a, b \in \mathbb{N}$, a stable matching M of $\mathcal{I}$is said to be an (a, b)-supermatch if for any set $\Psi \subseteq M$ of non-fixed stable pairs, where $|\Psi| = a$, there exists a stable matching M' such that $M' \cap \Psi = \emptyset$ and $d(M, M') \leq b + a$.*

The intractability result of the RSM is lifted to the RSR as the SR is a generalisation of the SM.

Theorem 1. *RSR is $\mathcal{NP}$-hard.*

Proof. The proof is straightforward as it is possible to create an SR instance I_{SR} from any given SM instance I_{SM} with the exact same stable matchings in polynomial-time by padding every other person of the same sex to the preference list of each person (see Lemma 1). Every (a, b)-supermatch in the I_{SM} is also an (a, b)-supermatch in the I_{SR} and vice versa. Hence, RSR is $\mathcal{NP}$-hard because RSM is $\mathcal{NP}$-hard [6]. □

3 Verification of $(1, b)$-supermatch in Polynomial Time

We prove in this section that checking if a stable matching M is a $(1, b)$-supermatch can be done in polynomial time. Indeed, we show in Theorem 3 how to construct the closet matching to M if any non-fixed pair in M wants to leave.

In order to show our main result, we first prove in Theorem 2 that any non-fixed pair can be:(1) produced by a unique rotation and eliminated by another one; or (2) eliminated by two different rotations and produced by two others (see later Example 1). In the first case, we shall denote by ρ_e the elimination rotation and by ρ_p the production rotation. In the second case, we shall denote by ρ_{p1}, ρ_{p2} the two production rotations and by ρ_{e1}, ρ_{e2} the two elimination rotations.

We assume w.l.o.g that the input instance admits at least two stable matchings. For any non-fixed stable pair (p_i, p_j), there are two possible cases to consider:

Case 1: (A) $f_{T_S}(i) = p_j$ and $l_{T_S}(j) = p_i$, or (B) $l_{T_S}(i) = p_j$ and $f_{T_S}(j) = p_i$;
Case 2: Otherwise.

Case 1 is a special case indicating that if one of the persons in the pair is the other ones' most preferred person in T_S (respectively, the other one is the least preferred person in T_S). Note that, in both cases $L_{T_S}(i) > 1$ and $L_{T_S}(j) > 1$, because the pairs are non-fixed. Later, we refer to these cases for identifying scenarios. In Lemma 3, we show how to identify the elimination rotation(s) for a given pair regardless of its case.

Lemma 3. *A non-fixed stable pair $\{p_i, p_j\}$ is eliminated by a rotation ρ if and only if $(p_i, p_j) \in \rho$ or $(p_j, p_i) \in \rho$.*

Proof. $\rightarrow$ Let $\rho = (x_0, y_0), (x_1, y_1) \ldots, (x_{|\rho|-1}, y_{|\rho|-1})$ be a rotation that eliminates $\{p_i, p_j\}$. Observe first that ρ is non-singular (otherwise $\{p_i, p_j\}$ is not stable). Recall that the elimination of ρ from a table T means for each pair $(x_m, y_m) \in \rho$, the deletion of $\{x_m, y_m\}$ and all pairs $\{y_m, z\}$ such that y_m prefers x_{m-1} to z from T. Table 2 gives an illustration of the preferences of x_m and y_m. The eliminating ρ moves x_m from y_m to y_{m+1} and deletes some $\{y_m, z\}$. In a similar way, eliminating $\bar{\rho}$ moves y_m from x_{m-1} to x_m and deletes some $\{x_m, z'\}$. Since every close complete subset contains either ρ or $\bar{\rho}$, then any pair $\{y_m, z\}$ and $\{x_m, z'\}$ cannot be part of any solution. Therefore, if ρ eliminates $\{p_i, p_j\}$ and $\{p_i, p_j\} \notin \rho$ then $\{p_i, p_j\}$ is not stable. This contradicts the fact that our pair $\{p_i, p_j\}$ is a non-fixed stable pair.

Table 2. An illustration of the preferences

p	Preference lists
$\ldots$	$\ldots$
x_m	$\ldots, y_m, z', y_{m+1} \ldots$
$\ldots$	$\ldots$
y_m	$\ldots, x_{m-1}, z, x_m, \ldots$
$\ldots$	$\ldots$

$\leftarrow$ By the definition of eliminating a rotation ρ from a table T , where $(p_i, p_j) \in \rho$, the elimination results in the deletion of p_j from p_i's list. Similarly, if $(p_j, p_i) \in \rho$ then it results in the deletion of p_i from p_j's list. □

Lemma 4 identifies the production rotations.

Lemma 4. *If a non-fixed stable pair $\{p_i, p_j\}$ is eliminated by ρ_e, then $\{p_i, p_j\}$ is produced by the dual of it, $\rho_p = \bar{\rho_e}$.*

Proof. A rotation is said to produce $\{p_i, p_j\}$ if eliminating it from a table T reduces $L_{T/\rho}(i)$ to a single entry, namely to p_j and $L_{T/\rho}(j)$ to p_i. We prove the existence of the production rotations over the two cases (Case 1 and Case 2) identified above.

We have two sub-cases in Case 1. First case is when $f_{T_S}(i) = p_j, l_{T_S}(j) = p_i$. In order to reduce p_i's list to only p_j, we need a rotation that moves p_i from his/her second best choice up to the first choice. We refer to this operation as *limiting p_i from right*. Similarly, to reduce the p_j's list to only p_i, we need a rotation that moves p_j from his/her second least-preferred person to the least preferred person. We refer to this operation as *limiting p_j from left*. Referring back to Table 2 for notation, the production rotation ρ_p of the pair $\{p_i, p_j\} = (x_m, y_m)$ must contain the pair $(y_{m+1}, x_m) \in \rho_p$ to limit x_m from right. Additionally, it must contain (y_m, x_{m-1}) to limit y_m from left. To illustrate, the production rotation has the shape: $\rho_p = \ldots, (y_m, x_{m-1}), (y_{m+1}, x_m), \ldots$. Note that, each ordered pair can only appear in exactly one rotation. Observe that, the dual of ρ_p contains the pair (x_m, y_m) by definition of dual. By Lemma 3, we know that the rotation

that contains (x_m, y_m) is the elimination rotation of the pair $\{p_i, p_j\}$. Therefore, $\rho_p = \bar{\rho_e}$ The proof for the second sub-case is similar, where $(y_m, x_m) \in \rho_e$.

For a pair $\{p_i, p_j\}$ of Case 2, each person has both more and less preferred people in their lists. Therefore, in order to produce a pair, their lists must be limited from both left and right. Let ρ_{p1} denote the rotation that limits p_i from left and p_j from right, and ρ_{p2} denote the rotation that limits p_i from right and p_j from left, respectively. Let the preference lists for the pair $\{p_i, p_j\}$ denoted by $L_{T_S}(i) = [\ldots, y_{m-1}, y_m, y_{m+1}]$ and $L_{T_S}(j) = [\ldots, x_{m-1}, x_m, x_{m+1}]$ where $\{p_i, p_j\} = (x_m, y_m)$. The pair (x_m, y_{m-1}) must be in ρ_{p1} to limit p_i from left and (x_{m+1}, y_m) be in ρ_{p1} to limit p_j from right. Additionally, the pair (y_{m+1}, x_m) must be in ρ_{p2} to limit p_i from right and (y_m, x_{m-1}) to limit p_j from left. Note that, the dual of ρ_{p1} contains (y_m, x_m), the dual of ρ_{p2} contains (x_m, y_m) by the definition of a dual rotation. By Lemma 3, we know these rotations are elimination rotations of the pair $\{p_i, p_j\}$.

Note that the two rotations ρ_{p1} and ρ_{p2} do not require one of them to be eliminated from the table first; they are incomparable. Therefore, depending on the order of elimination, both of them are production rotations. □

We sum up the findings above for the non-fixed stable pairs. If a pair is of Case 1, then there exists only one elimination rotation for this pair and only one production rotation as the dual of the elimination one. Because the preference list needs to be limited in only one direction. However, for the pairs of Case 2, there exist two elimination rotations for this pair, and also two other production rotations. Observe that, for each non-fixed stable pair $\{p_i, p_j\}$ in a stable matching M, the corresponding complete closed subset of M contains all production rotations of $\{p_i, p_j\}$. It is important to note that, especially for the pairs of Case 2, including one production rotation in the complete closed subset and not the other one, results in producing other partners for that pair. Subsequently, Theorem 2 is an immediate result of Lemmas 3 and 4.

Theorem 2. *Let $\{p_i, p_j\}$ be a non-fixed stable pair. If $\{p_i, p_j\}$ is of **Case 1**, then there exists a unique elimination rotation ρ_e, where $(p_i, p_j) \in \rho_e$ or $(p_j, p_i) \in \rho_e$, and a unique production rotation ρ_p, where $\rho_p = \bar{\rho_e}$. Otherwise (**Case 2**), there exist two different elimination rotations ρ_{e1} and ρ_{e2}, where $(p_i, p_j) \in \rho_{e1}, (p_j, p_i) \in \rho_{e2}$ and two rotations $\rho_{p1} = \bar{\rho_{e1}}, \rho_{p2} = \bar{\rho_{e2}}$ that produce the pair.*

Let S_P denote the set of all the complete closed subsets for the underlying SR instance. Lemma 5 gives a characterisation for the complete closed subsets.

Lemma 5. *Let $S \in S_P$. For each sink rotation ρ of S, the set $S \setminus \{\rho\} \cup \{\bar{\rho}\} \in S_P$.*

Proof. By definition of closed subset, every predecessor $\rho' \in N^-(\rho)$ is in S. Since ρ is a sink rotation, any successor $\rho^* \in N^+(\rho)$ is not in S. Therefore, by definition of the complete closed subset, we have $\bar{\rho^*} \in S$ and $\bar{\rho}$ is not in S. Using Lemma 2, we know that $\bar{\rho^*} \prec \bar{\rho}$. Hence, all predecessors of $\bar{\rho}$ are already in S, making $\bar{\rho}$ a neighbour rotation and results in $S \setminus \{\rho\} \cup \{\bar{\rho}\} \in S_P$. □

The distance between two stable matchings $d(M, M')$ is previously defined in Section 2 as the number of different pairs between M and M'. Observe that the distance can be calculated by also using their corresponding complete closed subsets. If $S \setminus S' = \{\rho\}$,

it means $\rho \in S$ and $\bar{\rho} \in S'$. We know that, $X(\{\rho\}) = Y(\{\bar{\rho}\})$ and $Y(\{\rho\}) = X(\{\bar{\rho}\})$. Therefore, between M and M', only the people in ρ (or $\bar{\rho}$) have different partners. This can also be generalised to a set of rotations. Hence, the distance can also be denoted as $d(S, S') = |X(S \setminus S') \cup Y(S \setminus S')|/2$. Note that $d(S', S) = d(S, S')$.

Lemma 6 identifies the closest stable matching to a stable matching M, when a rotation from its corresponding complete closed subset is to be removed.

Lemma 6. *Given a stable matching M and its corresponding complete closed subset S, if $\rho \in S$ is a rotation to remove from S, the closest stable matching M' to M such that $\rho \notin S'$ is found by the formula*[4]*:*

$$C(S, \rho) = S' = (S \setminus (\{\rho\} \cup N^+(\rho))) \cup \{\bar{\rho}\} \cup \bigcup_{\rho^* \in N^+(\rho)} \bar{\rho^*} \tag{1}$$

Proof. The proof of the defined set S' being a complete closed subset is obvious by using Lemma 2 and Lemma 5 as flipping a sink rotation of S yields in another complete closed subset. However, if ρ is not a sink rotation in S, we must flip all the successors of ρ to obtain a complete closed subset.

Let M^* denote the stable matching after flipping $\rho \in S$. Then, $d(M, M^*) = d(S, S^*) = |X(\{\rho\}) \cup Y(\{\rho\})|/2$. Now, let M^* denote the stable matching after flipping both $\rho, \sigma \in S$. Then, $d(M, M^*) = |X(\{\rho\}) \cup Y(\{\rho\}) \cup X(\{\sigma\}) \cup Y(\{\sigma\})|/2$. Observe that, flipping more rotations can only increase the distance between matchings. In Formula 1, the required number of flips is minimum. Therefore the function $C(S, \rho)$ returns the closest stable matching to M when $\rho \in S$ to be removed from S. □

Finally, Theorem 3 concludes how to find the closest stable matching M' to M if $\{p_i, p_j\} \in M$ wants to leave the M.

Theorem 3. *Given a stable matching M and a pair $\{p_i, p_j\}$ to leave M, the closest stable matching M' to M is identified by its corresponding S' using the Formula 1 as follows:*

1. *If Case 1, then $S' = C(S, \rho_p)$.*
2. *If Case 2, let M_1 and M_2 be the two stable matchings s.t. $S_1 = C(S, \rho_{p1})$ and $S_2 = C(S, \rho_{p2})$. Then $S' = S_1$ if $d(M, M_1) < d(M, M_2)$, otherwise $S' = M_2$.*

Proof. The proof is immediate from Theorem 2 and Lemma 6. □

In order to verify if a given M is a $(1, b)$-supermatch, all closest stable matchings to the given stable matching are found under the assumption that each non-fixed pair wants to leave the stable matching, one at a time. For each pair, its production rotation is identified and then Theorem 3 is applied to find the closest stable matching. Among all the closest stable matchings, the matching that results in the maximum distance to M sets the robustness of M, i.e. $b = d(M, M') - 1$, where 1 denotes the pair that wants to leave.

[4] The parentheses are used to indicate priority.

Example 1. Computing robustness Let us calculate the closest matching to M_6 given in Table 1. In Table 3, we identify the cases, and the production/elimination rotation(s) for assuming each pair leaves the M_6 at a time, and we apply Theorem 3 to find the robustness. The pair that has the maximum cost to be repaired sets the robustness value of the matching. Therefore, for this case, the robustness of M_6 is 3.

Table 3. Computing the closest matching to M_6

$\{p_i, p_j\}$	Case	ρ_p	ρ_e	$C(S, \rho)$	S	$d(M, M')$	S'	b
$\{p_1, p_7\}$	1	$\rho_p = \bar{\rho_6}$	$\rho_e = \rho_6$	$\{\rho_3, \rho_4, \rho_5, \rho_6, \rho_7\}$	S_4	4	S_4	3
$\{p_2, p_3\}$	2	$\rho_{p1} = \rho_7$	$\rho_{e1} = \bar{\rho_7}$	$\{\rho_3, \rho_4, \rho_5, \bar{\rho_6}, \bar{\rho_7}\}$	S_7	2	S_7	1
		$\rho_{p2} = \rho_3$	$\rho_{e2} = \bar{\rho_3}$	$\{\bar{\rho_3}, \rho_4, \rho_5, \rho_6, \rho_7\}$	S_1	4		
$\{p_4, p_9\}$	1	$\rho_p = \bar{\rho_6}$	$\rho_e = \rho_6$	$\{\rho_3, \rho_4, \rho_5, \rho_6, \rho_7\}$	S_4	4	S_4	3
$\{p_5, p_{10}\}$	2	$\rho_{p1} = \rho_4$	$\rho_{e1} = \bar{\rho_4}$	$\{\rho_3, \bar{\rho_4}, \rho_5, \bar{\rho_6}, \bar{\rho_7}\}$	S_2	3	S_2	2
		$\rho_{p2} = \bar{\rho_6}$	$\rho_{e2} = \rho_6$	$\{\bar{\rho_3}, \rho_4, \rho_5, \rho_6, \rho_7\}$	S_1	4		
$\{p_6, p_8\}$	1	$\rho_p = \rho_7$	$\rho_e = \bar{\rho_7}$	$\{\rho_3, \rho_4, \rho_5, \bar{\rho_6}, \bar{\rho_7}\}$	S_7	2	S_7	1

The production and elimination rotations of each pair can be identified in a preprocessing step. We show that checking if a stable matching is a $(1, b)$-supermatch can be performed in $O(n \times |\mathcal{V}|)$ time after the $O(n^3 log(n))$ preprocessing step for an instance where $2 \times n$ people are involved. The preprocessing step consists in identifying the rotations and building the reduced rotation poset ($O(n^3 logn)$) [7]; identifying all the predecessors and successors of each rotation ρ ($O(|\mathcal{V}|^2)$); and identifying elimination and production rotations for each pair $\{p_i, p_j\}$ whenever applicable in ($O(n^2)$). Given a stable matching M, its corresponding complete closed subset S is found by finding and adding the production rotation(s) of each pair and their predecessors into S by starting from an empty set ($O(n \times |\mathcal{V}|)$). Conversely, given a closet complete S, M can be constructed by eliminating all the rotations in S from T_S by respecting their precedence order. The order is found by applying sorting ($O(|\mathcal{V}| \times log|\mathcal{V}|)$). The main algorithm is to compute for each pair in M, the closest stable matching M' by using Theorem 3. Observe that computing the distance between two stable matchings takes $O(n)$ time and flipping a rotation takes a constant time. Moreover, the worst case of finding the closest stable matching is to flip all the non-singular rotations in S, where the number of all non-singular rotations is $|\mathcal{V}|/2$. Therefore, this computation takes $O(|\mathcal{V}|)$ time.

4 Finding Robust Solutions to the SR

We consider in this section two meta-heuristic approaches to solve the problem of finding a $(1, b)$-supermatch to a given Stable Roommates instance that minimizes the value of b.

4.1 Local Search

Considering the structural similarities between the RSM and the RSR, we tailored the local search model (LS) for the RSM, as it is shown that the LS model produces near

optimal solutions for RSM and is better than the proposed genetic algorithm [4]. In the generic LS model, there exists a neighbourhood $\boldsymbol{N}$ for the current solution. The algorithm works by searching the neighbourhood of the current solution, finding the best neighbour M_n in the neighbourhood and then proceeding the search by checking the neighbourhood of M_n. The aim is to find the stable matching that has the minimum b value. The search is restarted by a random stable matching at every few iterations to avoid getting stuck at a local optimum. The search continues until a termination criterion is met.

In our model, we have four termination criteria. The first one is a cut-off limit lim_{cutoff}, which "counts" the number of steps since the last best solution is found. The second one is the depth limit lim_{depth}, which indicates the depth of the neighbourhood search starting from a random stable matching. Another criterion is the optimality opt, which indicates if the algorithm has already found a solution with $b = 1$. Finally, we use a time limit lim_{time} for each instance.

The procedure starts by creating a random stable matching M_c as follows. We first mark all the non-singular rotations as available. Let A denote the set of rotations that are available. Then, we randomly select a rotation ρ from A and add it to the initially empty S_c. Subsequently, we remove ρ and $\bar{\rho}$ from A. We also add all predecessors ρ' of ρ that are not in S_c to S_c and remove ρ' and $\bar{\rho'}$ from A. This operation operates in a loop until $|S_c| = |\mathcal{V}|/2$. Once the complete closed subset S_c is found, its corresponding stable matching is computed by eliminating all rotations in S_c from T_S by respecting their precedence order.

After creating a random stable matching M_c, the neighbourhood $\boldsymbol{N}$ of M_c is found by checking all the sink rotations in S_c. By using Lemma 5, we know that flipping any sink rotation in M_c creates another stable matching M_n, which we refer as a *neighbour* of the M_c. The general procedure is the same as the one developed for the RSM [5]. In brief, the process starts by descending from the M_c by finding $\boldsymbol{N}$ of M_c. The next iteration descends from the neighbour of M_c that has the lowest b value. This loop is restarted every lim_{depth} iteration by a random M_c. The stable matching that has the minimum value of b as found during the search is returned as the solution.

The complexity of the LS procedure depends on the computation of the b values. Finding neighbours is based on the identification of the sink rotations of S_c, where there can be at most $|\mathcal{V}|/2$ sink rotations and then a constant cost for flipping each sink rotation. The best neighbour is identified after computing b values of $|\boldsymbol{N}|$ stable matchings. This procedure takes $O(k \times n \times |\mathcal{V}| \times |\boldsymbol{N}|)$, where k is the number of iterations and n is the number of non-fixed people.

4.2 Genetic Local Search (Hybrid)

Combining different search techniques to enhance the performance of a single model is proven to improve solution quality and the models [13,18]. Genc et al. propose three different models (constraint programming, local search and genetic algorithm) for finding $(1, b)$-supermatches to the RSM in [4]. The results indicate that genetic algorithm (GA) procedure has poor performance when compared to the LS. In this work, we consider combining the two metaheuristics: the genetic algorithm and the local search to provide a hybrid procedure. We denote this hybrid model as HB. The overview of the

GA procedure we use in the HB model is the same as the one used for RSM (details can be found in [5]).

The procedure begins by initialising a population of random stable matchings. Then, the population is evolved by randomly selecting individuals from the population, applying crossover, searching for neighbours of the products of crossover, applying mutation. This process is repeated until some termination criteria is met (no improvement, time-limit exceeded, optimal solution found). The procedure below gives a pseudo-code of the evolution phase of HB.

```
1:  procedure EVOLUTION()
2:      M_1 ← SELECTION()
3:      M_2 ← SELECTION()
4:      if M_1 ≠ M_2 then
5:          (M_c1, M_c2) ← CROSSOVER(M_1, M_2)
6:          N ←FINDNEIGHBOURS(M_c1)
7:          M_c1 ← BEST(N)
8:          N ←FINDNEIGHBOURS(M_c2)
9:          M_c2 ← BEST(N)
10:         REFINE(M_c1, M_c2)
11:         EVALUATION()
12:     M_fit ← GETFITTEST(P)
13:     M_m ← SELECTION()
14:     rand ← RANDOM(0, 1)
15:     if M_m ≠ M_fit and rand < p_m then
16:         MUTATION(M_m)
```

As can be seen from the procedure, the only LS enhancement to the GA algorithm is the search for the neighbours of the stable matchings after crossover (see Lines 6-9). Let M_{c1}, M_{c2} be the two stable matchings produced by the crossover. We update M_{c1} by its best neighbour after the neighbour search (same applies to the M_{c2}). Creating a random stable matching and finding neighbours are already discussed in Section 4.1.

If the original methods from LS and GA as described in [5], where the evolution phase is updated with the one here are used, we obtain the HB model for the RSM. In the RSR model, only the crossover and mutation operations are different than the original GA model defined for RSM. Instead of defining the crossover by adding rotations to the closed subset or removing them as we did for the RSM, we use the terminology *flip* for the RSR. Considering the Lemma 6, we define the crossover procedure for two stable matchings M_1, M_2 as follows. First, we find a random rotation $\rho_1 \in S_1$, and a random rotation $\rho_2 \in S_2$. If $\rho_1 \notin S_2$, then $\bar{\rho_1} \in S_2$ due to the completeness property of the closed subsets in SR. Therefore, we flip $\bar{\rho_1}$ in S_2 and the duals of all of its predecessors $\rho' \in N^-(\rho)$ if ρ' is not included in S_2. We apply the same procedure to the other stable matching as well. Moreover, for the mutation operation, we select a random rotation ρ from the reduced rotation poset of the underlying instance and also a stable matching M. If $\rho \in S$, we flip ρ and all the required predecessors. If its dual $\bar{\rho} \in S$, then we flip $\bar{\rho}$ and the predecessors.

5 Experiments

In this section, we first compare the performances of the HB and the LS proposed for the RSR. Then, we investigate the robustness of different sets of RSM and RSR instances [5]. The code is implemented in Java, reusing the RSM experiments from [4]. All experiments are performed on a Dell M600 with 2.66 Ghz processors under Linux, using three different randomisation seeds and fixing time limit $lim_{time} = 20$ mins, number of iterations without improvement $lim_{cutoff} = 10000$, the number of stable matchings in LS that descend from a random stable matching $lim_{depth} = 50$. We use the population size for the HB as $|\boldsymbol{P}| = 30$ and the mutation probability as $p_m = 0.7$. We use a high p_m as GA is suffering from getting stuck at local minima and randomisation helps with it. We discuss the size of the population for HB later.

HB v. LS for the RSR. Our first experiment is about the comparison of LS and HB models. Random SR instances have only a small number of stable matchings as we verify on the dataset RANDOM later [16]. For the comparison of HB and LS models, we look for instances that are likely to contain many stable matchings to gain more insight on their performances. For this purpose, we first created a dataset of purely random SM instances as each SM instance contains at least one stable matching, then we converted these instances to the SR. This conversion tackles the problem of random SR instances having only a few stable matchings, while preserving the randomness. Our SM dataset consists of 30 random instances of each size $n \in \{100 \times k \mid k \in \{1, \ldots, 10\}\}$. Note that, the resulting SR instances have size $2 \times n$.

Fig. 3 and Fig. 4 provide detail on the comparison between the LS and the HB. [6] Fig. 3 compares the average minimum b value found by the two models for each instance in the set. In the x-axis, the range shows the size of the instances such that all the instances that have x-values between $[0, 200]$ is of size 200, $[201 - 400]$ is of size 400, etc. We confirm by our experiments and also observe in Fig. 4 that for each instance that has size $200 \leq n \leq 600$, both models complete the search within the given time limit. Additionally, they either produce similar results (b values) or HB performs slightly better as can be observed in Fig. 3. The reason for exceeding the time limit in Fig. 4 is due to us not interrupting the construction of a stable matching. The construction of a stable matching consists of exposing all rotations in its complete closed subset in order starting from T_S. Then, the b value is computed. For large stable matchings, this computation becomes very costly. We can conclude that for small instances, both HB and LS perform well in terms of finding solutions with low b values. If the time is essential, HB model can be preferred over LS as it converges faster. Additionally, HB is able to find better solutions for larger instances.

Random RSM v. Random RSR. Next, we perform some tests for SM-SR comparison on our dataset RANDOM. Our dataset RANDOM consists of 30 purely randomly created SM and SR instances for each size $n \in \{100 \times k \mid k \in \{1, \ldots, 10\}\}$. Note that, for an SM instance of size n, there exists n men and n women in the problem. We have $2 \times n$

[5] Our datasets are publicly available at: github.com/begumgenc/rsmData

[6] The reader is referred to the online version for coloured version.

Fig. 3. Avg min b value found by LS and HB.

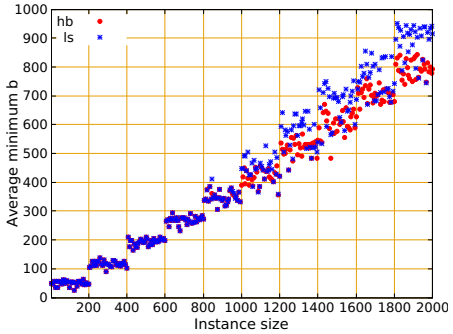

Fig. 4. Average time spent by LS and HB.

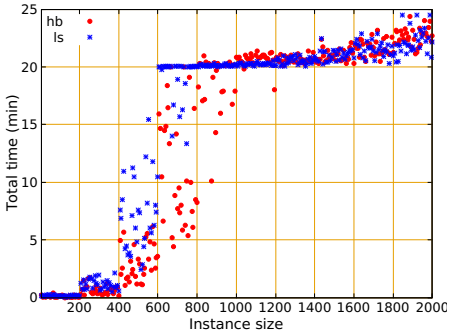

Table 4. Results on uniformly random instances for RSM.

n	$\|\mathcal{V}\|$	**sm**	**np**	**b**	**ratio**	t_{best}
100	22.02	47.9	75.12	48.27	0.64	0.02
200	41.59	116.9	166.19	115.34	0.69	0.10
300	60.22	182.4	263.94	193.08	0.73	0.37
400	74.51	244.1	356.58	265.98	0.74	0.77
500	91.47	322.5	456.00	350.16	0.76	2.18
600	103.82	394.9	551.10	425.51	0.77	3.67
700	117.08	449.6	646.69	505.61	0.78	5.89
800	131.81	527.6	749.98	595.64	0.79	9.09
900	146.34	585.5	848.32	679.82	0.80	14.60
1000	156.00	632.4	943.23	758.82	0.80	21.16

Table 5. Results on uniformly random instances for RSR.

n	$\|\mathcal{V}\|$	**sm**	**np**	**b**	**ratio**	t_{best}
100	3.91	3.78	17.91	5.31	0.3	0.003
200	3.87	3.94	26.76	8.52	0.32	0.003
300	4.36	4.56	35.53	11.22	0.32	0.017
400	4.71	5.92	37.64	10.93	0.29	0.048
500	4.29	4.81	37.62	11.70	0.31	0.066
600	4.16	4.48	42.44	14.47	0.34	0.130
700	4.58	5.50	48.71	16.02	0.33	0.312
800	4.93	5.99	55.02	17.39	0.32	0.498
900	4.82	7.07	57.64	18.50	0.32	0.662
1000	4.60	5.19	55.16	18.36	0.33	0.557

people in the corresponding SR instance. However, both have n pairs. All SR instances in RANDOM have at least two stable matchings. Considering the good performance of LS for small instances, we used the LS models for both RSM and RSR.

Table 4 and Table 5 present a summary of the robustness of random RSM and RSR instances. The columns report for each size the average value of: the total number of pairs in the instance (**n**), the number of rotations in the rotation poset or the reduced rotation poset ($|\mathcal{V}|$), the number of different stable matchings found during the search of LS (**sm**), the number of non-fixed pairs (**np**), the b value of the solution found (**b**), the ratio $\frac{b}{np}$ (**ratio**), and the time spent until finding the best solution by LS in seconds (t_{best}).

Observe from the tables that the random RSM instances contain many more stable matchings than the random RSR instances of similar sizes. Recall that, the value of sm denotes only the number of a subset of the stable matchings found during the search. However, we can confirm the RSR instances not containing many stable matchings by looking at the number of rotations in their rotation posets. Note that, for the RSR instances, when 1000 pairs are included, the corresponding rotation posets, on the average, contain $|\mathcal{V}| \approx 5$ rotations. This is mainly caused by the large numbers of

Table 6. Summary of the results on large instances for RSM.

instance			**LS**		**HB**, $\|P\| = 10$		**HB**, $\|P\| = 60$	
n	**np**	$\|\mathcal{V}\|$	**b**	**t (min)**	**b**	**t (min)**	**b**	**t (min)**
16	15.99	100.43	1.12	0.003	1.21	0.003	1.1	0.004
32	31.99	447.26	1.03	0.054	1.30	0.024	1.04	0.029
64	64	1889.95	1.685	3.158	1.74	0.824	1.28	0.916
128	128	7788.02	14.055	8.367	1.02	13.989	1.01	17.609

fixed-pairs in the random RSR instances. For instance, the average number of non-fixed pairs in the RSM instances of size 100 is $np = 75.12$. However, we observe in the large RSR instances that contain 1000 pairs that there are only 55.16 non-fixed pairs on the average, which is less than the smallest sized RSM instances that we tested. Note that, we measure the robustness ratio over the non-fixed pairs of the instances. It is desirable to obtain a smaller value for the ratio to indicate a better robustness for the instance. Because a smaller ratio indicates that a smaller proportion of the people that have alternative partners need to change their partners for a repair. Observe that, the ratio of the RSR instances is lower when compared to RSM. The ratio shows that the breakage of the pairs in the RSR instances are less costly to be repaired. Thus, we conclude that purely random RSR instances require a smaller proportion of the people to change their partners in the case of a breakage, when compared to the RSM.

Large RSM and RSR instances. In this experiment, we search for instances with potentially many number of stable matchings and low b values. Therefore, we generated a dataset called MANY consisting of 100 SM instances for each size $n = \{16, 32, 64, 128\}$ using the family described by Irving and Leather [11], and then used in [17]. Note that, each SM instance in this set has a corresponding SR instance (see Lemma 1), where the corresponding SR instance has a reduced rotation poset of twice the size of the rotation poset of the SM instance. First, let us introduce this family of instances described by Irving and Leather. They prove that any instance in the original family contains at least 2^{n-1} stable matchings for an instance of size $n = 2^i$. They define this family over two matrices for the preferences of each gender, and the preference lists of these large instances are obtained recursively by appending the following matrices until the desired instance size is found. In our dataset MANY, we slightly modify each instance of this original family by first randomly selecting two random men m_i, m_j. Then, we modify m_i's preference list by swapping the positions of two randomly selected women within the list. We repeat the same for m_j. We also modify the preference lists of two random women in the same way. In other words, the original preference set between the original and the modified instances have a Hamming Distance of 8.

Table 6 reports for each size the average value of: the number of all men or women (**n**), the number of non-fixed men(**np**), the number of rotations in the rotation poset ($|\mathcal{V}|$). Additionally, it reports the average minimum b found by the model LS, HB where population size $|P| = 10$, and HB where population size $|P| = 60$ (**b**); followed by the total time spent in minutes for each of the three models (**t (min)**).

This dataset shows that the robustness of instances that have many stable matchings is very high (i.e the value of b is low). For each instance size, our best model for that size is able to find solutions whose average b values evaluate to a b value that is $opt = 1 < b < 2$. For instance, for size $n = 16$, the LS model finds solutions such that for the breakage of any man on the solution, on the average, 1.12 other men need to break-up from their current partners. Similarly, for size $n = 128$, HB models find that the solution is guaranteed to be repaired by only 1.02 additional men's break-up.

As one can observe from Table 6, we ran the HB model by using different sizes of population. Observe that, reducing the number of individuals in the population of HB (60 to 10) causes the algorithm to find slightly worse solutions (i.e. larger b). For instance, for size $n = 64$, the average minimum b is found as 1.74 by a population of size 10, and 1.28 by a population of size 60. This is due to having an increased chance of getting stuck at local minima for a smaller population. On the other hand, LS finds competitive values for b for sizes $16 \leq n \leq 64$. However, as we can see for $n = 128$, LS finds solutions that are far away from the optimal solution. We conclude that, an improvement for HB by changing population size is possible in exchange of obtaining slightly worse solutions. LS performs well for smaller instances.

Recall that, each SM instance in MANY has a corresponding SR instance that has exactly the same stable matchings. We do not run the RSR models on this dataset as they are much slower. However, this test provides an insight to some RSM and RSR instances that are repairable at low additional costs.

6 Conclusions

We study the notion of (a, b)-supermatch in the context of Stable Roommates problem. We propose a polynomial-time algorithm based on the reduced rotation poset to verify if a stable matching is a $(1, b)$-supermatch. Next, we use this procedure to design local search (LS) and hybrid genetic local search (HB) models to find robust solutions for the $(1, b)$ case (i.e., (1,b)-supermatch with (possibly) the minimum b). We empirically show that the HB model usually performs better than LS for RSR. Furthermore, we perform an RSM/RSR comparison and identify a family of instances that are rich in stable matchings and very robust.

7 Acknowledgement

This material is based upon works supported by the Science Foundation Ireland under Grant No. 12/RC/2289 which is co-funded under the European Regional Development Fund.

References

1. Itai Ashlagi and Yannai A. Gonczarowski. Dating strategies are not obvious. *CoRR*, abs/1511.00452, 2015.
2. Haris Aziz, Péter Biró, Serge Gaspers, Ronald de Haan, Nicholas Mattei, and Baharak Rastegari. Stable matching with uncertain linear preferences. *CoRR*, abs/1607.02917, 2016.

3. Joanna Drummond and Craig Boutilier. Elicitation and approximately stable matching with partial preferences. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, pages 97–105. AAAI Press, 2013.
4. Begum Genc, Mohamed Siala, Barry O'Sullivan, and Gilles Simonin. Finding robust solutions to stable marriage. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 631–637, 2017.
5. Begum Genc, Mohamed Siala, Barry O'Sullivan, and Gilles Simonin. Finding robust solutions to stable marriage. *CoRR*, abs/1705.09218, 2017.
6. Begum Genc, Mohamed Siala, Gilles Simonin, and Barry O'Sullivan. Complexity study for the robust stable marriage problem. *Theoretical Computer Science*, 2019.
7. Dan Gusfield and Robert W. Irving. *The Stable Marriage Problem: Structure and Algorithms*. MIT Press, Cambridge, MA, USA, 1989.
8. Emmanuel Hebrard. *Robust solutions for constraint satisfaction and optimisation under uncertainty*. PhD thesis, University of New South Wales, 2007.
9. Alan Holland and Barry O'Sullivan. Super solutions for combinatorial auctions. In *Recent Advances in Constraints, Joint ERCIM/CoLogNet International Workshop on Constraint Solving and Constraint Logic Programming, CSCLP, 2004, Lausanne, Switzerland, June 23-25, 2004, Revised Selected and Invited Papers*, pages 187–200, 2004.
10. Robert W. Irving. An efficient algorithm for the "stable roommates" problem. *J. Algorithms*, 6(4):577–595, 1985.
11. Robert W. Irving and Paul Leather. The complexity of counting stable marriages. *SIAM J. Comput.*, 15(3):655–667, 1986.
12. Royi Jacobovic. Perturbation robust stable matching. *CoRR*, abs/1612.08118, 2016.
13. Antoon Kolen and Erwin Pesch. Genetic local search in combinatorial optimization. *Discrete Applied Mathematics*, 48(3):273 – 284, 1994.
14. Dmitry Lebedev, Fabien Mathieu, Laurent Viennot, Anh-Tuan Gai, Julien Reynier, and Fabien De Montgolfier. On using matching theory to understand p2p network design. In *INOC 2007, International Network Optimization Conference*, 2007.
15. B. Lussier, R. Chatila, F. Ingrand, M.O. Killijian, and D. Powell. On fault tolerance and robustness in autonomous systems, 2004.
16. Boris Pittel. The" stable roommates" problem with random preferences. *The Annals of Probability*, pages 1441–1477, 1993.
17. Mohamed Siala and Barry O'Sullivan. Rotation-based formulation for stable matching. In *Proceedings of CP'17, Melbourne, Australia, September*, pages 262–277, 2017.
18. Nico L. J. Ulder, Emile H. L. Aarts, Hans-Jürgen Bandelt, Peter J. M. van Laarhoven, and Erwin Pesch. Genetic local search algorithms for the traveling salesman problem. In Hans-Paul Schwefel and Reinhard Männer, editors, *Parallel Problem Solving from Nature*, pages 109–116, Berlin, Heidelberg, 1991. Springer Berlin Heidelberg.