A Quadratic Lower Bound for Stable Roommates Solvability

Will Rosenbaum
School of Computer Science and Informatics, University of Liverpool, UK

Abstract

In their seminal work on the Stable Marriage Problem (SM), Gale and Shapley introduced a generalization of SM referred to as the Stable Roommates Problem (SR). An instance of SR consists of a set of $2n$ agents, and each agent has preferences in the form of a ranked list of all other agents. The goal is to find a one-to-one matching between the agents that is stable in the sense that no pair of agents have a mutual incentive to deviate from the matching. Unlike the (bipartite) stable marriage problem, in SR, stable matchings need not exist. Irving devised an algorithm that finds a stable matching or reports that none exists in $O(n^2)$ time. In their influential 1989 text, Gusfield and Irving posed the question of whether $\Omega(n^2)$ time is required for SR solvability – the task of deciding if an SR instance admits a stable matching.

In this paper, we show that any (randomized) algorithm that decides SR solvability requires $\Omega(n^2)$ adaptive Boolean queries to the agents' preferences (in expectation). Our argument follows from a reduction from the communication complexity of the set disjointness function. The query lower bound implies quadratic time lower bounds for Turing machines, and memory access lower bounds for random access machines. Thus, we establish that Irving's algorithm is optimal (up to a logarithmic factor) in a very strong sense.

2012 ACM Subject Classification Theory of computation → Communication complexity; Theory of computation → Problems, reductions and completeness; Theory of computation → Algorithmic game theory; Mathematics of computing → Discrete mathematics

Keywords and phrases Stable Roommates, Stable Matching, Lower Bound, Communication Complexity

Digital Object Identifier 10.4230/LIPIcs.ESA.2026.36

Acknowledgements I thank Christine Cheng for numerous discussions of the Stable Roommates Problem, David Manlove for his helpful commentary on an earlier draft of this manuscript, and the anonymous reviewers for their thoughtful suggestions for improvement.

1 Introduction

Since its formalization by Gale and Shapley in 1962 [6], the Stable Marriage Problem (SM) and its variants has become a central topic in economics, computer science, and mathematics. The rich theoretical landscape of stable allocation problems and their diverse applications gained further widespread recognition with the 2012 Nobel Prize in Economics, which was awarded to Roth and Shapley for their contributions to this field. In Gale and Shapley's original formulation of the problem, an SM instance consists of two disjoint sets of agents, traditionally referred to as men and women. Each agent has preferences in the form of a ranked list of agents of the opposite gender. The goal is to find a marriage (i.e., a one-to-one correspondence) between men and women that is stable in the sense that no man-woman pair have a mutual incentive to deviate from the marriage. The seminal result of Gale and Shapley proves that a stable marriage always exists and describes an efficient algorithm for finding one.

In [6], Gale and Shapley also describe a generalization of SM in which there is no bipartition of the agent set into men and women. In this variant, each agent ranks all other agents – not just those of the opposite gender. Again, the goal is to find a stable matching: a one-to-one correspondence between agents such that no pair has a mutual incentive to deviate from their assigned partners (see Section 2.1 for formal definitions). This generalization of SM is referred to as the Stable Roommates Problem (SR).

Since its introduction by Gale and Shapley, SR has been the subject of a large volume of theoretical work and SR has found many applications. As the problem's name suggests, SR naturally models situations in which "peers" should be matched with one another, such as identifying roommates in housing searches [18]. Variants of SR can be used to model pair-wise kidney exchanges for organ donation [20]. The general study of this problem has led to the development of algorithms to facilitate kidney exchange markets for organ donation in several countries [1, 23, 4]. SR algorithms have also been employed for peer-to-peer file sharing networks [14, 16], and forming pairs of players in chess tournaments [12]. We refer the readers to [15] for a thorough overview of SR and applications.

Unlike the bipartite SM – where stable marriages are guaranteed to exist for all preferences – Gale and Shapley observed that SR instances need not admit stable matchings (see Section 2.1.1 for an explicit example). The question of whether or not a given instance admits a stable matching is known as the SR solvability problem. Gale and Shapley [6] and subsequently Knuth [11] asked whether there exists an efficient algorithm for SR solvability (and finding a stable matching if one exists), where Knuth suggested the problem might be NP-complete. In 1985, Irving [9] devised an efficient algorithm for SR solvability. Irving's algorithm finds a stable matching or reports that none exists in $O(n^2)$ time on instances with $2n$ agents.[1] In 1990, Ng and Hirschberg [17] showed that $\Omega(n^2)$ time is necessary to find a stable marriage (in the bipartite stable marriage problem). Their result implies the same lower bound holds for *finding* a stable matching for SR, but the lower bound does not apply to the decision problem of determining whether or not an SR instance admits a stable matching. In principle, it could be possible that unsolvable SR instances contain some "forbidden structure" whose existence can be detected in sub-linear time in the instance size. In their influential 1989 text on stable matching problems, Gusfield and Irving [8] listed finding a lower bound (or $o(n^2)$ time algorithm) for deciding SR solvability as one of 12 open questions for future research. This question was again listed as an open problem in Manlove's comprehensive 2012 text on algorithmic aspects of stable matchings [15]. To our knowledge, the question has remained open since.

1.1 Our Contributions

Our main result is to prove an $\Omega(n^2)$ lower bound for any (randomized or deterministic) algorithm deciding SR solvability. While Gusfield and Irving's question is phrased in terms of running time lower bounds for a particular representation of the agents' preferences, we give a more general result that bounds the number of Boolean queries necessary to decide SR solvability.

Much of the historical work on SM and SR assumes that the input to these problems is presented as ordered lists of agents' preferences, where each agent has a unique ID. Random access to an instance is thus naturally modeled via adaptive queries of the form "What agent has rank r in agent a's preference list?" which are typically treated as unit-cost queries. Similarly, if ranking arrays are also provided, it is natural to assume unit cost queries of

[1] Observe that since the input consists of $2n$ lists each of length $2n-1$, $O(n^2)$ is *linear* in the input size.

the form "What is agent b's rank in agent a's preference list?" The lower bounds of Ng and Hirschberg [17] (cf. [8, Section 1.5]) show that $\Omega(n^2)$ such queries are necessary to find a stable matching.[2]

In this work, we make no assumptions about how agents' preferences are presented. Instead, we assume preferences are accessed via Boolean queries, i.e., an adaptive sequence of yes/no questions about the agents' preferences. While each query in [17, 8] only references a single agents' preferences, we allow for queries that depend on multiple agents' preferences. More precisely, we assume that the set A of $2n$ agents is partitioned into fixed sets $A_1, A_2, \ldots, A_k$ where each part A_i contains at most $n/2$ agents, and each query concerns only the preferences of agents in one A_i. For example, this allows for queries of the form "Does agent a prefer agent b to agent c?" from which we can recover an agent's preference list in $O(n \log n)$ queries. Additionally, our model allows for more general queries such as "Is there any agent in A_i that prefers agents in B to agents in C?" where B and C are any subsets of agents. Thus, our computational model allows us to consider mechanisms that *elicit* information about agents' preferences rather than assuming that agents' preferences are already presented as, e.g., ordered lists. This more general framework of adaptive queries allows us to model scenarios in which preferences are not initially known, perhaps even to the agents themselves, and we seek to devise a mechanism (e.g., through questionnaires) that elicits sufficient information about the agents' preferences to determine if the instance is solvable. Our main result shows that any such mechanism that decides SR solvability requires $\Omega(n^2)$ Boolean queries in the worst case.

▶ **Theorem 1** (Informal, c.f. Theorem 4). *Any algorithm that decides SR solvability requires $\Omega(n^2)$ Boolean queries to the agents' preferences for instances with $2n$ agents. This lower bound applies to randomized protocols (in expectation) and allows for arbitrary Boolean queries made to individual agents' preferences as well as queries made to predetermined batches of agents of size at most $n/2$ (i.e., queries that involve more than one agent's preferences).*

Since all preferences can be inferred using $O(n^2 \log n)$ Boolean queries, the lower bound of Theorem 1 is tight up to a factor of $O(\log n)$. Assuming agents are given IDs of $O(\log n)$ bits, each query in the preference models of [17, 8] can be simulated using $O(\log n)$ Boolean queries (e.g., "What is the ith bit of the ID of the agent that has rank r in a's preference list?"). Theorem 1 therefore implies that $\Omega(n^2/\log n)$ queries are necessary to decide SR solvability in their preference model. Thus, this result settles Gusfield and Irving's original question, up to a factor of $O(\log n)$.

One advantage of phrasing our lower bound in terms of (arbitrary) Boolean queries is that the lower bound extends to many models of computation simultaneously and it is agnostic to the representation of the preferences.

▶ **Corollary 1.1.** *The following lower bounds hold for deciding SR solvability of instances with $2n$ agents:*

1. *Any (multi-tape, probabilistic) Turing machine that decides SR solvability requires $\Omega(n^2)$ time (in expectation).*
2. *Any (randomized) random access machine (RAM) with word size $O(\log n)$ bits that decides SR solvability requires $\Omega(n^2/\log n)$ memory accesses (in expectation).*

[2] Ng and Hirschberg's result [17] proves this lower bound for SM, but as SR is a generalization of SM, the result holds *a fortiori* to SR as well.

The lower bounds of 1 and 2 hold even if the agents' preferences are preprocessed arbitrarily in batches of size up to $n/2$ (where separate batches are preprocessed independently of each other).

Another interpretation of our lower bound is for mechanisms that decide SR solvability by eliciting agents' *implicit* preferences. In practice, agents' preferences may not be known explicitly, even to the agents themselves.[3] Thus, it is desirable to design a mechanism (such as questionnaires) that elicits sufficient information about the agents' preferences to determine SR solvability. Theorem 1 implies that any such mechanism requires $\Omega(n^2)$ Boolean queries. Thus, determining SR solvability is essentially as hard as learning the agents' preferences.

In order to prove Theorem 1, we employ a reduction from the two-party communication complexity of the set disjointness function, disj. This technique was previously applied by Gonczarowski et al. [7] to obtain lower bounds for the (bipartite) SM. Given the framework of query lower bounds via communication complexity [2, 5], our construction and argument are quite simple. The basic idea is to define a family of SR instances that correspond to inputs to the disjointness function. That is, given an input (x, y) to disj, we define an SR instance $R(x, y)$ such that $R(x, y)$ admits a stable matching if and only if $\text{disj}(x, y) = 1$. We then argue that the correspondence $(x, y) \mapsto R(x, y)$ has the property that any algorithm that decides SR solvability using q queries implies the existence of a communication protocol for disj using q bits of communication. Our main result follows from well-known communication complexity lower bounds for disj [10, 19]. We describe this technique more thoroughly in Section 2.2 before defining the reduction formally in Section 3.

▶ **Remark 1.2.** *Our $\Omega(n^2)$ query lower bound is only tight to Irving's algorithm [9] up to a logarithmic factor. This is because an SR instance of size $2n$ contains $2n$ lists of $2n - 1$ agents. If each agent's identity is encoded with $\Theta(\log n)$ bits, the total size of preference lists is $\Theta(n^2 \log n)$ bits. Irving's algorithm allows for access to a single entry on a preference list as a unit cost operation, whereas this operation would correspond to $\Theta(\log n)$ Boolean queries in our model. Thus, in our model, Irving's algorithm uses $\Theta(n^2 \log n)$ Boolean queries. We leave it as an open question whether the lower bound can be improved to $\Omega(n^2 \log n)$ Boolean queries (i.e., a lower bound that is truly linear in the instance size) or if the logarithmic factor in the running time can be improved.*

1.2 Related Work

SR was introduced by Gale and Shapley [6] who showed that SR instances need not admit stable matchings. Knuth [11] explicitly posed the question of whether there is an efficient algorithm to find a stable matching or report that none exists. Irving [9] devised an algorithm that finds a stable matching or correctly determines that none exists in $O(n^2)$ time (assuming each preference list entry can be accessed in $O(1)$ time). Gusfield and Irving's influential book [8] gives a comprehensive overview of early algorithmic work on SR, including structural aspects of the set of stable matchings for an SR instance. Additionally, [8] listed two questions related to efficiently deciding SR solvability. The first question asked if it is possible to

[3] For example, consider the eponymous application of determining suitable roommates for housing. The question of determining if a prefers potential roommate b to c may rely on knowing a significant amount of information about b and c that is not initially known to a. In cases of assigning university housing (where potential roommates may not know each other beforehand) responses to questionnaires are often used as a proxy for explicit preferences, with the hope that responses to the questions elicit sufficient information about the "true" preferences to determine roommate compatibility.

generate a "succinct certificate" for non-solvability of an SR instance. This question was positively answered by Tan [22] who demonstrated that a "stable partition" (of size $O(n)$) can be computed in $O(n^2)$ time. Given the partition, solvability of an SR instance can be verified in $O(n)$ time. Gusfield and Irving's second question asked if it is possible to decide SR solvability in $o(n^2)$ time directly. Our lower bound gives a negative answer to this question. Given Tan's result, our lower bound also implies that finding a stable partition requires $\Omega(n^2)$ time. Several previous works [17, 3, 21, 7] proved $\Omega(n^2)$ lower bounds for *finding* stable marriages in SM (and *a fortiori* for finding stable matchings in SR), but none of these results imply lower bounds for the decision problem of SR solvability. We refer the reader to the text of Manlove [15] for an account of more recent developments related to SR.

Our main lower bound result follows from a reduction from the two-party communication complexity of the disjointness function (see Section 2.2 for a formal definition). The two-party communication complexity model was introduced by Yao in [24]. The (randomized) two-party communication complexity of the disjointness function was first characterized by Kalyanasundaram and Schintger [10], and subsequently a simpler argument was found by Razborov [19].

Most closely related to the present paper is the work of Gonczarowski et al. [7]. In [7], the authors apply the communication complexity of set disjointness to obtain lower bounds for finding and verifying (bipartite) stable marriages. Indeed, our techniques in this paper closely follow that of [7].[4] Reductions from communication complexity to obtain (sublinear time) lower bounds were employed by Blais, Brody, and Matulef [2] for property testing. The technique was further codified in the work of Eden and Rosenbaum [5] who proved a variety of lower bounds for sublinear time algorithms. Our description of our lower bound in terms of an "embedding" of set disjointness follows the general approach and vocabulary of [5].

2 Background and Definitions

2.1 The Stable Roommates Problem

Formally, an instance of the Stable Roommates problem (hereafter, SR), consists of a set S of $2n$ ***agents*** together with ***preferences*** for each agent in the form a ranked list of all other agents. We say that a ***prefers*** agent b to c if b appears before c on a's preference list. A matching $M = \{\{a_1, b_1\}, \{a_2, b_2\}, \ldots, \{a_n, b_n\}\}$ is a partition of S into subsets of size 2. Given an SR instance and a matching M, we say that a pair $\{a_i, b_j\} \notin M$ is a ***blocking pair*** if a_i prefers b_j to b_i and b_j prefers a_i to a_j. A matching that does not induce any blocking pairs is said to be ***stable***.

An SR instance is ***solvable*** if it admits a stable matching. The ***SR Solvability*** problem is the problem of deciding if a given SR instance is solvable.

An efficient ($O(n^2)$ time) algorithm for solving SR Solvability was proposed by Irving in [9]. We will not require a description of the full algorithm, but our analysis will rely on an initial processing of the preference lists from Irving's algorithm. Our terminology and exposition of Irving's algorithm follow Gusfield and Irving's text [8, Chapter 4].

The algorithm maintains a ***preference table*** that initially lists all of the agents' preferences. As the algorithm proceeds, pairs of agents that cannot appear in any stable matching are removed. When a pair $\{a, b\}$ is removed from the table, a is removed from b's preference

[4] An anonymous reviewer for a previous version of this paper pointed out that Theorem 1 can also be derived by suitably modifying one of the constructions from [7]. This connection was not known to the authors of [7], nor is it clear if the modified construction is simpler than one presented here. Thus, our goal here is to provide a simple and self-contained proof of Theorem 1.

list and b is removed from a's preference list, while the relative order of other agents in the table are unchanged. The initial phase of Irving's algorithm, which we refer to as the Phase 1 algorithm (Algorithm 1) proceeds as follows. Each agent is initially "free." Free agents are selected in an arbitrary order to propose to the next remaining agent on their preference list. When a proposes to b, a becomes ***semiengaged*** to b and a is no longer free. Upon receiving a proposal from a, b rejects any agent currently semiengaged to b, and all pairs of agents $\{a', b\}$ such that b prefers a to a' are removed from the preference table. This process continues until either all agents are semiengaged or all free agents have empty preference lists.

Algorithm 1 Phase 1 Algorithm [9, 8].

```
1:  procedure PhaseOne
2:      Assign each agent to be free
3:      while Some free agent x has a nonempty list do
4:          y ← first agent on x's preference list          ▷ x proposes to y
5:          if some z is semiengaged to y then
6:              assign z to be free                          ▷ y rejects z
7:          end if
8:          assign x to be semiengaged to y
9:          for all successors x' of x on y's list do
10:             remove {x', y} from the preference table
11:         end for
12:     end while
13: end procedure
```

Irving showed that when the Phase 1 algorithm terminates, the resulting preference table has the following properties.

▶ **Lemma 2.1** (Irving [9], c.f. [8, Section 4.2.2])**.** *Consider the preference table resulting from an execution of Algorithm 1 on an SR instance. Then:*

1. *The resulting preference table is independent of the order in which free agents propose ([8, Lemma 4.2.1]).*
2. *Any pair $\{a, b\}$ that is not contained in the table is not contained in any stable matching. In particular, if any preference list in the table is empty, then the instance is not solvable ([8, Lemma 4.2.3]).*
3. *If M is a matching such that all pairs in M appear in the preference table, then no pair $\{a, b\}$ not appearing the preference table can block M. In particular, if the preference table consists only of a matching M, M is the unique stable matching for the instance (c.f., [8, Lemmas 4.2.1 and 4.2.4]).*

In our lower bound construction we will use only Lemma 2.1 to argue the (non)solvability of the required SR instances.

2.1.1 An Unsolvable SR Instance

In order to motivate the construction used to achieve our main result, it is helpful to understand the following non-solvable instance of SR.

▶ **Example 2.2.** *Consider agent set* $S = \{a, b, c, d\}$ *with the following preferences*

agent	*preferences*		
a	b	c	d
b	c	a	d
c	a	b	d
d	*arbitrary*		

To see why this instance is not solvable, observe that agents a*,* b*, and* c *form a "directed triangle" where they all rank one another in the top two, but the most preferred agents form a cycle* $a \to b \to c \to a$*. Thus in any matching* M*,* d*'s partner in* M *will form a blocking pair with one of the other two matched agents. For example, if* $\{a, d\} \in M$*, then* $\{a, c\}$ *forms a blocking pair.*

The idea of our lower bound construction is to define a family of SR instances in which there are many possible sub-instances homomorphic to Example 2.2. For this family, determining SR solvability is equivalent to finding if a given instance contains such a sub-instance. We argue that determining the existence of such a sub-instance requires $\Omega(n^2)$ accesses to the agent's preferences using a reduction from communication complexity.

2.2 Communication Complexity and Embeddings

Our proof of Theorem 1 employs a reduction from two-party communication complexity [24]. In this computational model, the goal is to compute a value $f(x, y)$ where f is some Boolean function, and $x, y \in \{0, 1\}^N$ are inputs. The inputs are distributed between two parties, traditionally referred to as Alice and Bob. Alice knows only x, while Bob knows only y. The goal is for Alice and Bob to both learn the value of $f(x, y)$ while exchanging the fewest possible number of bits. The (deterministic) ***communication complexity*** of f is defined to be the minimum number of bits needed for the worst-case input for any communication protocol that computes f. Randomized communication complexity is defined analogously using randomized communication protocols that are allowed to err with some small probability. We refer the reader to the text of Kushilevitz and Nisan [13] for a formal description of the communication models and a thorough exposition of fundamental results.

In this paper, we focus on the communication complexity of the ***disjointness function*** defined by

$$\text{disj}(x, y) = \begin{cases} 1 & \text{if } \sum_{i=1}^{N} x_i y_i = 0 \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

The (randomized) communication complexity of the disjointness function was first characterized by Kalyanasundaram and Schintger [10], and a simpler argument was found by Razborov [19] soon after.

▶ **Theorem 2** ([10, 19])**.** *Any (deterministic or randomized) communication protocol for* disj *requires* $\Omega(N)$ *bits of communication. This lower bound holds if inputs are restricted to be either disjoint* ($\sum_i x_i y_i = 0$) *or uniquely intersecting* ($\sum_i x_i y_i = 1$)*.*

Our strategy in obtaining our main result is to define a reduction from disj to SR solvability. That is, we define a function that maps each input (x, y) for the two party disjointness function to an SR instance $R(x, y)$. This mapping $(x, y) \mapsto R(x, y)$ is an ***embedding*** of set disjointness in the sense defined by Eden and Rosenbaum [5]. This means that the function has the following properties:

1. $R(x, y)$ is solvable if and only if $\text{disj}(x, y) = 1$.
2. The response to any allowed Boolean query to $R(x, y)$ can be computed by Alice (who knows only x) and Bob (who knows only y) using $O(1)$ bits of communication.

Under a generalization of these conditions, Eden and Rosenbaum [5] showed that communication lower bounds imply query lower bounds. We emphasize that this technique yields adaptive query lower bounds: future queries are allowed to depend on the responses to previous queries. Specializing their result to our setting gives the following result.

▶ **Theorem 3** (c.f. [5]). *Suppose there exists an embedding from* disj *instances of size N to SR solvability instances of size n satisfying the conditions above. Then any randomized or deterministic protocol for SR solvability for instances of size n must use $\Omega(N)$ adaptive queries in expectation.*

We elaborate on what the "allowed queries" for our embedding are. In general one cannot allow for arbitrary queries made to arbitrary sets of agents. Indeed, this would allow queries such as "Is the instance solvable?" which trivially solves SR solvability with a single query. At the other extreme, one could restrict queries of particular forms such as "Does agent a prefer agent b to agent c?" We assume that there is an arbitrary fixed partition of the agents $A = A_1 \cup A_2 \cup \cdots \cup A_k$ where the A_i – referred to as ***batches*** – are pair-wise disjoint and each A_i satisfies $|A_i| \leq n/2$. The allowed queries are the set of Boolean queries where each query concerns the preferences of agents in some A_i. For any such partition, this allows for arbitrary Boolean queries to individual agents' preferences, but also queries whose responses depend on multiple agents within a batch, such as "Does any agent in batch A_i prefer agent b to agent c?"

3 The Lower Bound

In this section we give the main argument for our lower bound for SR solvability. As described above, the idea is to give an embedding of the two party set disjointness problem into SR. The embedding will be as follows. For any even positive integer n, we will construct SR instances with $2n$ agents. To this end, let $N = n^2/4$. To simplify notation in the construction, we index bits of the inputs x, y with two indices from the range $[n/2]$, i.e., $x = (x_{ij})$ with $1 \leq i, j \leq n/2$.

Given any pair of Boolean vectors x, y satisfying the disjoint or unique intersection promise (see the statement of Theorem 2), we will define an SR instance $R(x, y)$ as follows. $R(x, y)$ contains $2n$ agents partitioned into 4 sets, each of size $n/2$: $S = A \cup B \cup C \cup D$. We will write the agents in set A as $A = \{a_1, a_2, \ldots, a_{n/2}\}$, and similarly with B, C, and D. The preferences of the agents in set A are determined from x, and the preferences of agents of B are determined from y. The preferences of agents in sets C and D are fixed independent of x and y. The preferences are determined as follows.

- An agent $a_i \in A$ prefers in order
  1. all b_j with $x_{ij} = 1$ (arbitrary order)
  2. c_i
  3. all other agents in arbitrary order
- An agent $b_j \in B$ prefers in order
  1. all c_i with $y_{ij} = 1$ (arbitrary order)
  2. all a_i with $y_{ij} = 1$ (arbitrary order)
  3. d_j
  4. all other agents in arbitrary order
- An agent $c_i \in C$ prefers in order
  1. a_i
  2. all agents $b_j \in B$ in arbitrary order
  3. all other agents in arbitrary order
- An agent $d_j \in D$ prefers in order
  1. b_j
  2. all other agents in arbitrary order

We illustrate examples of disjoint (solvable) and uniquely intersecting (non-solvable) instances in Figures 1 and 2, respectively. The figures also illustrate the proof Proposition 3.1 from which our main results follow.

▶ **Proposition 3.1.** *Suppose* $x, y \in \{0,1\}^{n/2\times n/2}$*, and let* $R(x,y)$ *be the SR instance defined above.*

1. *If* $\mathrm{disj}(x,y) = 1$ *(i.e., x and y are disjoint), then* $R(x,y)$ *admits a unique stable matching* M*, namely*

$$M = \{\{a_i, c_i\} \mid i \in [n/2]\} \cup \{\{b_j, d_j\} \mid j \in [n/2]\} \tag{2}$$

2. *If x and y are uniquely intersecting (i.e.,* $\sum_{i,j} x_{ij}y_{ij} = 1$*), and hence* $\mathrm{disj}(x,y) = 0$*, then* $R(x,y)$ *is not solvable.*

In particular, the association $(x,y) \mapsto R(x,y)$ *is an embedding of disjointness into SR solvability.*

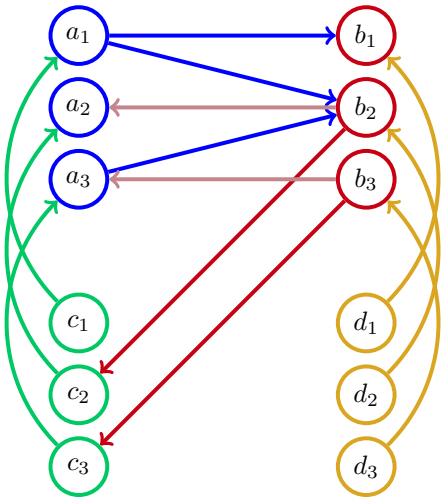
Proof. To prove the proposition, we consider an execution of the Phase 1 algorithm, Algorithm 1. We first consider the case where $\mathrm{disj}(x,y) = 1$ (case 1 of the proposition). Consider an execution of Algorithm 1 with the agents proposing in the following order. By Lemma 2.1, the resulting preference table does not depend on the order of the proposals. We illustrate an example of the execution of such an instance with $n = 6$ in Figure 1.

1. Agents in C propose. Specifically, each $c_i \in C$ proposes to a_i. Since a_i receives a proposal from c_i, in Line 10, only b_js with $x_{ij} = 1$ will remain on a's preference list (above c_i). Since $\mathrm{disj}(x,y) = 1$, $x_{ij} = 1$ implies $y_{ij} = 0$, hence all pairs $\{a_i, b_j\}$ with $y_{ij} = 1$ are removed.
2. Agents in D propose. Specifically, each $d_j \in D$ proposes to b_j. When b_j receives the proposal from d_j, only c_is with $y_{ij} = 1$ remain on b_j's preference list, as all a_i's with $y_{ij} = 1$ were removed after the proposals in Step 1. Since $\mathrm{disj}(x,y) = 1$, if $y_{ij} = 1$, then $x_{ij} = 0$. Therefore, all remaining pairs of the form $\{a_i, b_j\}$ are removed from the preference table.
3. Agents in A propose. At this point, each a_i's preference list consists of a single entry, c_i, so a_i proposes to c_i. Since a_i is first on c_i's preference list, all remaining pairs involving c_i are removed from the preference table. In particular, all pairs of the form $\{c_i, b_j\}$ are removed.
4. Agents in B propose. At this point, each b_j's preference list consists of a single entry, d_j, so b_j proposes to d_j.

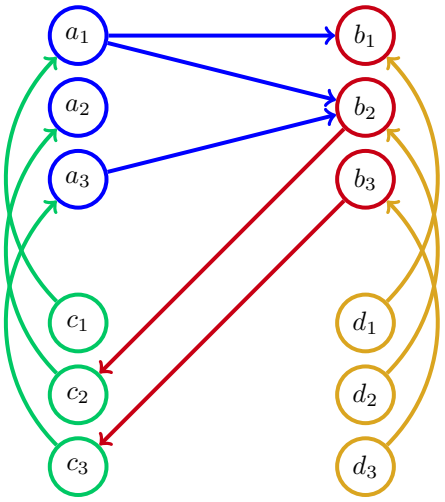
After agents from B propose, all agents are semiengaged so the algorithm terminates. The preference table consists only of the matching M in the proposition's statement, which is therefore a stable matching (by Lemma 2.1).

Now consider the case where x and y are uniquely intersecting. Let $k, \ell \in [n/2]$ be the indices for which $x_{k\ell} = y_{k\ell} = 1$, while $x_{ij}y_{ij} = 0$ for all $(i,j) \neq (k,\ell)$. As above, we consider an execution of Algorithm 1. We illustrate an example of the execution of such an instance in Figure 2.

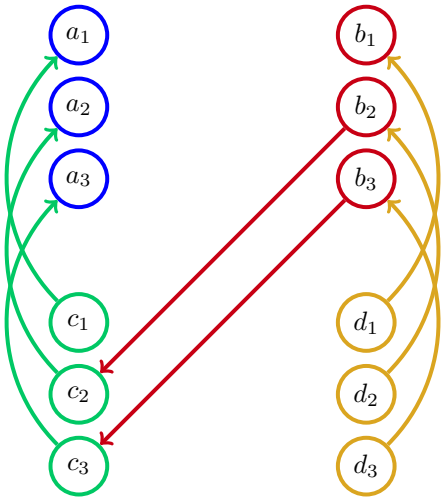
1. Agents in C propose. In this case, pairs $\{a_i, b_j\}$ with $y_{ij} = 1$ are removed from the preference table except for $\{a_k, b_\ell\}$. Further, all pairs $\{a_i, d_j\}$ are removed.
2. Agents in D propose. Now all pairs $\{a_i, b_j\}$ with $x_{ij} = 1$ are removed from the preference table except for $\{a_k, b_\ell\}$.
3. Agents in A propose. All a_i with $i \neq k$ propose to c_i, while a_k proposes to b_ℓ. Subsequently, b_ℓ rejects d_ℓ. As each agent c_i with $i \neq k$ receives a proposal from a_i, each $\{c_i, d_\ell\}$ is removed from the preference table.



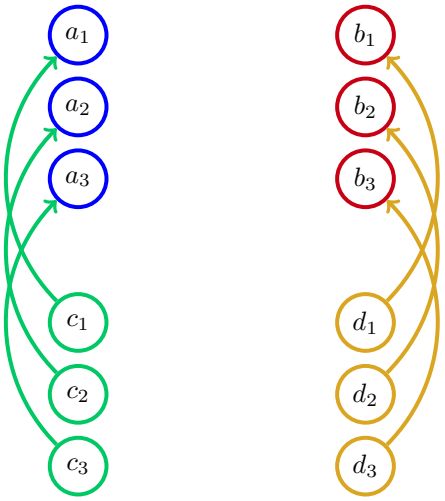
(a) Initial preferred pairs in the table.



(b) Pairs removed after proposals from C.

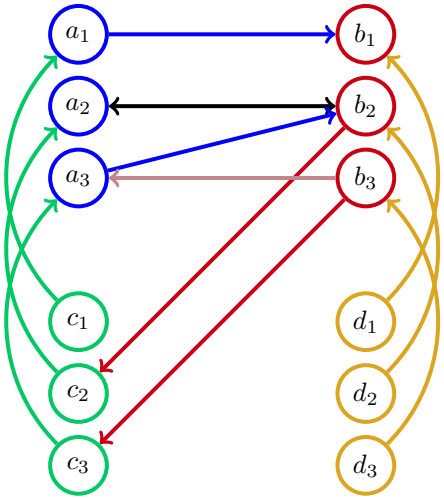


(c) Pairs removed after proposals from D.

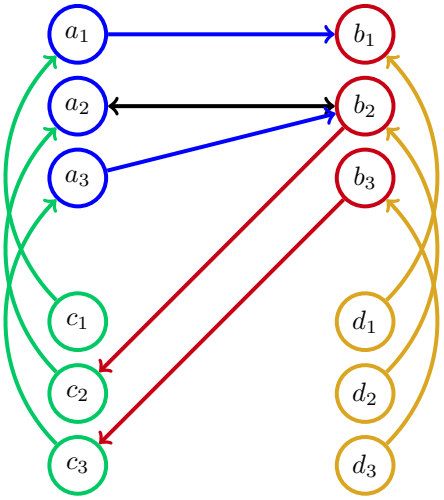


(d) Remaining pairs after proposals from A.

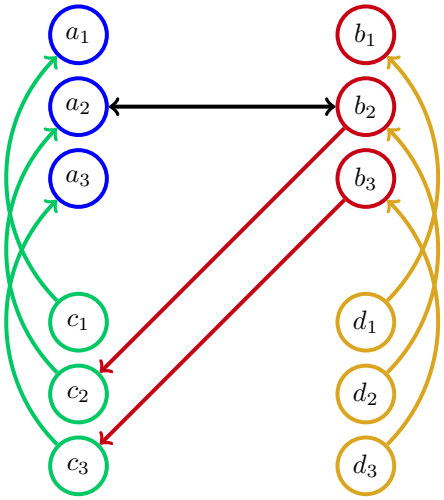
Figure 1 An illustration of the embedding of disjointness for $N = 3 \times 3$ and $n = 6$. This instance corresponds to $x_{1,1} = x_{1,2} = x_{3,2} = 1$, while the remaining values of $x_{ij} = 0$, and $y_{2,2} = y_{3,3} = 1$ with the remaining values of $y_{ij} = 0$. Thus, $\text{disj}(x, y) = 1$. The edges between agents are colored by agent preferences: the blue edges from the a_i correspond to their most preferred partners (preferred above c_i). Similarly, the red edges from the b_j correspond to their most preferred edges, while the tan edges from the b_j are second most-preferred group. The green edges from the c_i and orange edges from the d_j correspond to those agent's most preferred partners. Other possible partners are not depicted. Sub-figure (a) represents the remaining pairs in the preference table before the first rounds of proposals, while (b), (c), and (d) depict the remaining pairs after each round of proposals. The final figure depicts the unique stable matching for this instance.



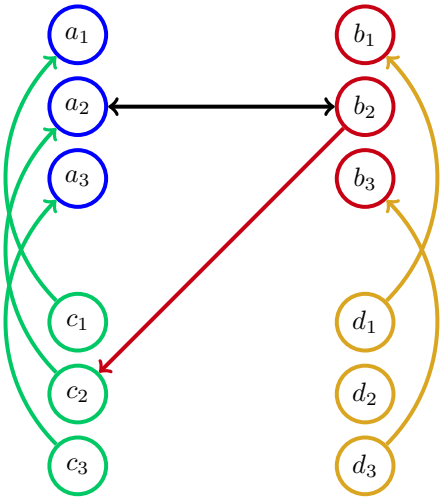
(a) Initial preferred pairs in the table.



(b) Pairs removed after proposals from C.



(c) Pairs removed after proposals from D.



(d) Remaining pairs after proposals from A.

Figure 2 An illustration of the embedding of disjointness for $N = 3 \times 3$ and $n = 6$. This instance corresponds to $x_{1,1} = x_{2,2} = x_{3,2} = 1$, while the remaining values of $x_{ij} = 0$, and $y_{2,2} = y_{3,3} = 1$ with the remaining values of $y_{ij} = 0$. Thus, $\text{disj}(x, y) = 1$ with $x_{2,2} = y_{2,2} = 1$. Sub-figure (a) represents the remaining pairs in the preference table before the first rounds of proposals. The black edge between a_2 and b_2 indicates that both agents rank each other highly: b_2 is a_2's most preferred choice, while b_2 only prefers c_2 to a_2. Figures (b), (c), and (d) depict the remaining pairs after each round of proposals. Note that in all of the figures, the pair $\{a_2, b_2\}$ is preferred by a_2 to c_2 and preferred by b_2 to d_2. Therefore, this edge is not removed after either the C proposals nor the D proposals in figures (b) and (c). Subsequently, when agents in A propose, a_2 proposes to b_2, after which b_2 rejects d_2. At this point d_2's preference list is empty, hence the instance does not admit a stable matching.

4. Agents in B propose. In this case, each b_j with $j \neq \ell$ proposes to d_j, while b_ℓ proposes to c_k. Subsequently, all pairs $\{d_j, d_\ell\}$ are removed from the preference table, as is the pair $\{c_k, d_\ell\}$.

At this point all pairs involving d_ℓ have been removed, so d_ℓ's preference list is empty. Therefore, by Lemma 2.1, $R(x, y)$ does not admit a stable matching. ◀

We now state and prove our main lower bound for the stable roommates problem. In the formal statement, we allow for any procedure that performs arbitrary adaptive Boolean queries to the agents' preference lists, and queries can even be made to "batches" of agents – i.e., a fixed partition of the agents – so long as no batch contains more than $n/2$ agents. For example, this allows queries of the form, "Does any agent in set A prefer agent b to b'?" Specifically, our argument holds for any Boolean query that does not involve agents from *both* sets A and B in the construction above. An upper bound on the batch size is necessary allowing a batch size of $2n$ would allow for the query "Does S admit a stable matching?" as a single query.

▶ **Theorem 4.** *Any randomized (or deterministic) mechanism that decides SR solvability on instances with n agents using adaptive Boolean query access to the agents' preferences requires $\Omega(n^2)$ Boolean queries in expectation. This bound applies even if queries can be made to fixed batches of agents of size up to $n/2$.*

Proof. Let $\mathcal{A}$ be any algorithm that determines SR solvability using q queries for SR instances of size n. Then we can use $\mathcal{A}$ to define a two party communication protocol for disj using the embedding of Proposition 3.1 as follows. Suppose Alice and Bob hold inputs $x, y \in \{0,1\}^{n/2 \times n/2}$, respectively, which are promised to be either disjoint or uniquely intersecting. Note that the set disjointness instance has size $N = n^2/4$. Alice forms preferences of agents in the set A as in the embedding, and Bob does the same for B. Thus, the instance has $2n$ agents in total. Note that the preferences of agents in C and D are independent of x and y, hence they are known to both Alice and Bob. Alice and Bob then simulate the algorithm $\mathcal{A}$: the response to any query that $\mathcal{A}$ makes to agents in $A \cup C \cup D$ can be computed by Alice, who then sends the Boolean result to Bob as a single bit. Similarly, the response to any query made to $B \cup C \cup D$ can be computed by Bob, who then sends the response to Alice. When $\mathcal{A}$ terminates after q queries, both Alice and Bob can compute the output of $\mathcal{A}$ from the communication transcript, hence they both know whether or not the $R(x, y)$ is solvable. By Proposition 3.1, this output determines $\text{disj}(x, y)$.

Since this communication protocol solves an arbitrary instance of disj (with the unique intersection promise) with input size $n^2/4$ using q bits of communication, we have $q = \Omega(n^2)$ by Theorem 2. ◀

Finally, we argue that the query lower bound of Theorem 4 implies the computational lower bounds listed in Corollary 1.1.

Proof of Corollary 1.1 (Sketch). The lower bound for Turing machines follows from the observation that each "read" operation performed by a Turing machine can be modelled as a response to a single Boolean query (the value of the bit that is read). Thus, the query lower bound implies that any Turing machine the decides SR solvability must read $\Omega(n^2)$ bits of its input, hence its running time is $\Omega(n^2)$.

Similarly, for random access machines (RAMs), each memory access to a word of size $O(\log n)$ bits can be simulated by $O(\log n)$ Boolean queries. Hence we obtain a $\Omega(n^2/\log n)$ lower bound on the number of memory accesses.

We observe that the arguments above make no reference to the representation of the input to the problem, and the query lower bound argument holds for any Boolean queries made to the input so long as a single query's response does not depend on the preferences of agents from both A and B. Following an arbitrary preprocessing of the agents' preferences where agents in A are processed independently from agents in B, each bit of the resulting encoding is determined by either preferences in $A \cup C \cup D$ or preferences in $B \cup C \cup D$. In the former case, reading a single bit of the processed input can be simulated with a single Boolean query to A (as preferences in C and D are fixed) in the former case and B in the latter case. Thus, the lower bounds apply to preprocessed inputs as well. ◀

References

1 Alliance for Paired Kidney Donation. Alliance for paired kidney donation. `https://www.paireddonation.org/`. Accessed: 2025-02-08.

2 Eric Blais, Joshua Brody, and Kevin Matulef. Property testing lower bounds via communication complexity. *computational complexity*, 21(2):311–358, 2012. `doi:10.1007/s00037-012-0040-x`.

3 Jen-Hou Chou and Chi-Jen Lu. Communication requirements for stable marriages. In *Proceedings of the 7th International Conference on Algorithms and Complexity (CIAC)*, pages 371–382, 2010. `doi:10.1007/978-3-642-13073-1_33`.

4 de Klerka Marry, Karin M. Keizer, Frans H.J. Claas, Marian Witvliet, Bernadette J.J.M. Haase-Kromwijk, and Weimar Willem. The dutch national living donor kidney exchange program. *American Journal of Transplantation*, 5(9):2302–2305, 2005. `doi:10.1111/j.1600-6143.2005.01024.x`.

5 Talya Eden and Will Rosenbaum. Lower Bounds for Approximating Graph Parameters via Communication Complexity. In Eric Blais, Klaus Jansen, José D. P. Rolim, and David Steurer, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2018)*, volume 116 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 11:1–11:18, Dagstuhl, Germany, 2018. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.APPROX-RANDOM.2018.11`.

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

7 Yannai A. Gonczarowski, Noam Nisan, Rafail Ostrovsky, and Will Rosenbaum. A stable marriage requires communication. *Games and Economic Behavior*, 118:626–647, 2019. `doi:10.1016/j.geb.2018.10.013`.

8 Dan Gusfield and Robert W Irving. *The Stable Marriage Problem: Structure and Algorithms*. MIT Press, 1989.

9 Robert W Irving. An efficient algorithm for the "stable roommates" problem. *Journal of Algorithms*, 6(4):577–595, 1985. `doi:10.1016/0196-6774(85)90033-1`.

10 Bala Kalyanasundaram and Georg Schintger. The probabilistic communication complexity of set intersection. *SIAM Journal on Discrete Mathematics*, 5(4):545–557, 1992. `doi:10.1137/0405044`.

11 Donald E. Knuth. *Marriage stables et leurs relations avec d'autres problèmes combinatoires*. Les Presses de l'Université de Montréal, 1976.

12 Eija Kujansuu, Tuukka Lindberg, and Erkki Mäkinen. The stable roommates problem and chess tournament pairings. *Divulgaciones Matemáticas*, 7(1):19–28, 1999.

13 Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, 1997.

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*, Spa, Belgium, 2007. URL: `https://hal.science/hal-00159678`.

15 David F Manlove. *Algorithmics of Matching Under Preferences*. WORLD SCIENTIFIC, 2013. doi:10.1142/8591.

16 Fabien Mathieu. *Acyclic preference-based systems*, pages 1165–1203. Springer US, Boston, MA, 2010. doi:10.1007/978-0-387-09751-0_42.

17 Cheng Ng and Daniel Hirschberg. Lower bounds for the stable marriage problem and its variants. *SIAM Journal on Computing*, 19(1):71–77, 1990. doi:10.1137/0219004.

18 Nitsan Perach, Julia Polak, and Uriel G. Rothblum. A stable matching model with an entrance criterion applied to the assignment of students to dormitories at the technion. *International Journal of Game Theory*, 36(3):519–535, 2008. doi:10.1007/s00182-007-0083-4.

19 Alexander A. Razborov. On the distributional complexity of disjointness. *Theoretical Computer Science*, 106, 1992. doi:10.1016/0304-3975(92)90260-M.

20 Alvin E. Roth, Tayfun Sönmez, and M. Utku Ünver. Pairwise kidney exchange. *Journal of Economic Theory*, 125(2):151–188, 2005. doi:10.1016/j.jet.2005.04.004.

21 Ilya Segal. The communication requirements of social choice rules and supporting budget sets. *Journal of Economic Theory*, 136:341–378, 2007. doi:10.1016/J.JET.2006.09.011.

22 Jimmy J.M Tan. A necessary and sufficient condition for the existence of a complete stable matching. *Journal of Algorithms*, 12(1):154–178, 1991. doi:10.1016/0196-6774(91)90028-W.

23 UK National Health Service. Uk living kidney sharing scheme. https://www.odt.nhs.uk/living-donation/uk-living-kidney-sharing-scheme/. Accessed: 2025-02-08.

24 Andrew Chi-Chih Yao. Some complexity questions related to distributive computing (preliminary report). In *Proceedings of the 11th Annual ACM Symposium on Theory of Computing (STOC)*, pages 209–213, 1979. doi:10.1145/800135.804414.