How good are Popular Matchings?

Krishnapriya A M[1]*, Meghana Nasre[2], Prajakta Nimbhorkar[3], Amit Rawat[4] **

[1] Citrix Research and Development India,
[2] Indian Institute of Technology Madras, India,
[3] Chennai Mathematical Institute India and UMI ReLaX
[4] University of Massachusetts Amherst, USA

Abstract. In this paper, we consider the Hospital Residents problem (HR) and the Hospital Residents problem with Lower Quotas (HRLQ). In this model with two sided preferences, stability is a well accepted notion of optimality. However, in the presence of lower quotas, a stable and feasible matching need not exist. For the HRLQ problem, our goal therefore is to output a *good* feasible matching assuming that a feasible matching exists. Computing matchings with minimum number of blocking pairs (Min-BP) and minimum number of blocking residents (Min-BR) are known to be NP-Complete. The only approximation algorithms for these problems work under severe restrictions on the preference lists. We present an algorithm which circumvents this restriction and computes a *popular* matching in the HRLQ instance. We show that on data-sets generated using various generators, our algorithm performs very well in terms of blocking pairs and blocking residents. Yokoi [20] recently studied *envy-free* matchings for the HRLQ problem. We propose a simple modification to Yokoi's algorithm to output a *maximal envy-free* matching. We observe that popular matchings outperform envy-free matchings on several parameters of practical importance, like size, number of blocking pairs, number of blocking residents.
In the absence of lower quotas, that is, in the Hospital Residents (HR) problem, stable matchings are guaranteed to exist. Even in this case, we show that popularity is a practical alternative to stability. For instance, on synthetic data-sets generated using a particular model, as well as on real world data-sets, a popular matching is on an average 8-10% larger in size, matches more number of residents to their top-choice, and more residents prefer the popular matching as compared to a stable matching. Our comprehensive study reveals the practical appeal of popular matchings for the HR and HRLQ problems. To the best of our knowledge, this is the first study on the empirical evaluation of popular matchings in this setting.

1 Introduction

In this paper, we study two problems – the Hospital Residents (HR) problem and the Hospital Residents problem with Lower Quotas (HRLQ). The input to the HR problem is a bipartite graph $G = (\mathcal{R} \cup \mathcal{H}, E)$ where $\mathcal{R}$ denotes a set of residents, $\mathcal{H}$ denotes a set of hospitals, and an edge $(r, h) \in E$ denotes that r and h are acceptable to each other. Each vertex has a *preference list* which is a strict ordering on its neighbors. Further, each hospital has a positive upper-quota $q^+(h)$. In the HRLQ problem, additionally, a hospital has a non-negative lower-quota $q^-(h)$. A *matching* M is a subset of E such that every resident is assigned at most one hospital and every hospital is assigned at most upper-quota many residents. Let $M(r)$ denote the hospital to which resident r is matched in M. Analogously, let $M(h)$ denote the set of residents that are matched to h in M. A matching M in an HRLQ instance is *feasible* if for every hospital h, $q^-(h) \leq |M(h)| \leq q^+(h)$. The goal is to compute a feasible matching that is *optimal* with respect to the preferences of the residents and the hospitals. When both sides of the bipartition express preferences, *stability* is a well-accepted notion of optimality. A stable matching is defined by the absence of a *blocking pair*.

Definition 1. *A pair $(r, h) \in E \setminus M$ blocks M if either r is unmatched in M or r prefers h over $M(r)$ and either $|M(h)| < q^+(h)$ or h prefers r over some $r' \in M(h)$. A matching M is* stable *if there does not exist any blocking pair w.r.t. M, else M is unstable.*

* Part of this work was done when the author was an M.Tech. student at IIT Madras.
** Part of this work was done when the author was an MS student at IIT Madras.

From the seminal result of Gale and Shapley [12], it is known that every instance of the HR problem admits a stable matching and such a matching can be computed in linear time in the size of the instance. In contrast, there exist simple instances of the HRLQ problem which do not admit any matching that is both *feasible and stable.*

$\mathbf{r_1} : h_1\ h_2$	$[\mathbf{0}, \mathbf{2}]$	$\mathbf{h_1} : r_1\ r_2\ r_3$
$\mathbf{r_2} : h_1\ h_2$	$[\mathbf{1}, \mathbf{1}]$	$\mathbf{h_2} : r_2\ r_1$
$\mathbf{r_3} : h_1$		

Fig. 1. A hospital residents instance G with $\mathcal{R} = \{r_1, r_2, r_3\}$ and $\mathcal{H} = \{h_1, h_2\}$. The quotas of the hospitals are $q^-(h_1) = 0, q^+(h_1) = 2, q^-(h_2) = q^+(h_2) = 1$. The preferences can be read from the tables as follows: r_1 prefers h_1 over h_2 and so on. The three matchings M_1, M_2 and M_3 are shown below. M_2 and M_3 are feasible but unstable since (r_2, h_1) blocks M_2 and (r_1, h_1) blocks M_3. $M_1 = \{(r_1, h_1), (r_2, h_1)\}$ $M_2 = \{(r_1, h_1), (r_2, h_2), (r_3, h_1)\}$ $M_3 = \{(r_1, h_2), (r_2, h_1), (r_3, h_1)\}$.

Figure 1 shows an HRLQ instance where h_2 has a lower-quota of 1. The instance admits a unique stable matching M_1 which is not feasible. The instance admits two maximum cardinality matchings M_2 and M_3, both of which are feasible but unstable. This raises the question what is an optimal feasible matching in the HRLQ setting?

Our goal in this paper is to propose *popularity* as a viable option in the HRLQ setting, and compare and contrast it by an extensive experimental evaluation with other approaches proposed for this problem. Before giving a formal definition of popularity, we describe some approaches from the literature to the HRLQ problem.

Hamada et al. [14] proposed the optimality notions (i) minimum number of blocking pairs (Min-BP) or (ii) minimum number of residents that participate in any blocking pair (Min-BR). Unfortunately, computing a matching which is optimal according to any of the two notions is NP-Hard as proved by Hamada et al. [14]. On the positive side, they give approximation algorithms for a special case of HRLQ, complemented by matching inapproximability results. Their approximation algorithms require that all the hospitals with non-zero lower-quota have *complete* preference lists (*CL-restriction*).

There are two recent works [18,20] which circumvent the CL-restriction and work with the natural assumption that the HRLQ instance admits some feasible matching. Nasre and Nimbhorkar [18] consider the notion of *popularity* and show how to compute a feasible matching which is a maximum cardinality popular matching in the set of feasible matchings. Yokoi [20] considers the notion of *envy-freeness* in the HRLQ setting. We define both popularity and envy-freeness below.

Popular matchings: Popular matchings are defined based on comparison of two matchings with respect to votes of the participants. To define popular matchings, first we describe how residents and hospitals vote between two matchings M and M'. We use the definition from [18]. For a resident r unmatched in M, we set $M(r) = \bot$ and assume that r prefers to be matched to any hospital in his preference list (set of acceptable hospitals) over $\bot$. Similarly, for a hospital h with capacity $q^+(h)$, we set the positions $q^+(h) - |M(h)|$ in $M(h)$ to $\bot$, so that $|M(h)|$ is always equal to $q^+(h)$. For a vertex $u \in \mathcal{R} \cup \mathcal{H}$, $vote_u(x, y) = 1$ if u prefers x over y, $vote_u(x, y) = -1$ if u prefers y over x and $vote_u(x, y) = 0$ if $x = y$. Thus, for a resident r, $vote_r(M, M') = vote_r(M(r), M'(r))$.

Voting for a hospital: A hospital h is assigned $q^+(h)$-many votes to compare two matchings M and M'; this can be viewed as one vote per position of the hospital. A hospital is indifferent between M and M' as far as its $|M(h) \cap M'(h)|$ positions are concerned. For the remaining positions of hospital h, the hospital defines a function $\mathbf{corr}_h$. This function allows h to decide any pairing of residents in $M(h) \setminus M'(h)$ to residents in $M'(h) \setminus M(h)$. Under this pairing, for a resident $r \in M(h) \setminus M'(h)$, $\mathbf{corr}_h(r, M, M')$ is the resident in $M'(h) \setminus M(h)$ corresponding to r. Then $vote_h(M, M') = \sum_{r \in M(h) \setminus M'(h)} vote_h(r, \mathbf{corr}_h(r, M, M'))$. As an example, consider $q^+(h) = 3$ and $M(h) = \{r_1, r_2, r_3\}$ and $M'(h) = \{r_3, r_4, r_5\}$. To decide its votes, h compares between $\{r_1, r_2\}$ and $\{r_4, r_5\}$ and one possible $\mathbf{corr}_h$ is to pair r_1 with r_4

and r_2 with r_5. Another $\mathbf{corr}_h$ function is to pair r_1 with r_5 and r_2 with r_4. The choice of the pairing of residents using $\mathbf{corr}_h$ is determined by the hospital h. With the voting scheme as above, popularity can be defined as follows:

Definition 2. *A matching M is more popular than M' if $\sum_{u \in \mathcal{R} \cup \mathcal{H}} vote_u(M, M') > \sum_{u \in \mathcal{R} \cup \mathcal{H}} vote_u(M', M)$. A matching M is popular if there is no matching M' more popular than M.*

It is interesting to note that the algorithms for computing popular matchings do not need the **corr** function as an input. The matching produced by the algorithms presented in this paper is popular for *any* **corr** function that is chosen.

Definition 3. *Given a feasible matching M in an* HRLQ *instance, a resident r has* justified envy *towards r' with $M(r') = h$ if h prefers r over r' and r is either unmatched or prefers h over $M(r)$. A matching is envy-free if there is no resident that has a justified envy towards another resident.*

Thus envy-freeness is a relaxation of stability. A matching that is popular amongst feasible matchings always exists [18], but there exist simple instances of the HRLQ problem that admit a feasible matching but do not admit a feasible envy-free matching. Yokoi [20] gave a characterization of HRLQ instances that admit an envy-free matching and an efficient algorithm to compute some envy-free matching.

An envy-free matching need not even be *maximal.* In the example in Figure 1 the matching $M = \{(r_2, h_2)\}$ is envy-free. Note that M is not maximal, and not even a *maximal envy-free matching.* The matching $M' = \{(r_1, h_1), (r_2, h_2)\}$ is a maximal envy-free matching, since addition of any edge to M' violates the envy-free property. The matchings M_2 and M_3 shown in Figure 1 are not envy-free, since r_2 has justified envy towards r_3 in M_2 whereas r_1 has a justified envy towards r_3 in M_3. The algorithm in [20] outputs the matching M which need not even be maximal; Algorithm 2 in Section 2.2 outputs M' which is guaranteed to be maximal envy-free. Finally, the Algorithm 1 in Section 2.1 outputs M_2 (see Figure 1) which is both maximum cardinality as well as popular.

Popularity is an interesting alternative to stability even in the absence of lower-quotas, that is, in the HR problem. The HR problem is motivated by large scale real-world applications like the National Residency Matching Program (NRMP) in US [3] and the Scottish Foundation Allocation Scheme (SFAS) in Europe [4]. In applications like NRMP and SFAS it is desirable from the social perspective to match as many residents as possible so as to reduce the number of unemployed residents and to provide better staffing to hospitals. Birò et al. [7] report that for the SFAS data-set (2006–2007) the administrators were interested in knowing if relaxing stability would lead to gains in the size of the matching output. It is known that the size of a stable matching could be half that of the maximum cardinality matching M^*, whereas the size of a maximum cardinality popular matching is at least $\frac{2}{3}|M^*|$ (see e.g. [16]). Thus, theoretically, popular matchings give an attractive alternative to stable matchings.

Popular matchings have gained lot of attention and there have been several interesting results in the recent past [6,8,10,9,16,18,19]. The algorithms used and developed in this paper, are inspired by a series of papers [9,11,16,18,19]. One of the goals in this paper is to complement these theoretical results with an extensive experimental evaluation of the quality of popular matchings. We now summarize our contributions below:

1.1 Our contribution

Results for HRLQ: Algorithm for popular matchings: We propose a variant of the algorithm in [18]. Whenever the input HRLQ instance admits a feasible matching, our algorithm outputs a (feasible) popular matching amongst all feasible matchings. We report the following experimental findings:

– Min-BP *and* Min-BR *objectives:* In all the data-sets, the matching output by our algorithm is at most 3 times the optimal for the Min-BP problem and very close to optimal for the Min-BR problem. In the HRLQ setting, no previous approximation algorithm is known for incomplete preference lists. The only known algorithms which output matchings

with theoretical guarantees for the Min-BP and Min-BR problems work under the CL-restricted model [14]. We remark that the algorithm of [18] can not provide any bounded approximation ratio for Min-BP and Min-BR problems as it may output a feasible matching with non-zero blocking pairs and non-zero blocking residents even when the instance admits a stable feasible matching.
- *Comparison with envy-free matchings:* Based on the algorithm by Yokoi [20], we give an algorithm to compute a maximal envy-free matching, if it exists, and compare its *quality* with a popular matching. Besides the fact that popular matchings exist whenever feasible matchings exist, which is not the case with envy-free matchings, our experiments illustrate that, compared to an envy-free matching, a popular matching is about 32%–43% larger, matches about 15%–38% more residents to their top choice and has an order of magnitude fewer blocking pairs and blocking residents.

Empirical Evaluation of known algorithms for HR : For the HR problem, we implement and extensively evaluate known algorithms for maximum cardinality popular matching and popular amongst maximum cardinality matching on synthetic data-sets as well as limited number of real-world data-sets.

- Our experiments show that a maximum cardinality popular matching, as well as popular amongst maximum cardinality matchings are 8–10% larger in size than a stable matching. We also observe that a maximum cardinality popular matching almost always fares better than a stable matching with respect to the number of residents matched to their rank-1 hospitals, and the number of residents favoring a maximum cardinality popular matching over a stable matching.

We note that these properties cannot be proven theoretically as there are instances where they do not hold. Despite these counter-examples (see Appendix 5) our empirical results show that the desirable properties hold on synthetic as well as real-world instances. Hence we believe that popular matchings are a very good practical alternative.
Organization of the paper: In Section 2.1, we present our algorithm to compute a matching that is popular amongst feasible matchings for the HRLQ problem. Our algorithm for computing a maximal envy-free matching is described in Section 2.2. In Section 3 we give details of our experimental setup and the data-generation models developed by us. Section 4 gives our empirical results for the HRLQ and HR problems. We refer the reader to [9,19] for known algorithms for computing popular matchings in the HR problem.

2 Algorithms for HRLQ problem

In Section 2.1, we present our algorithm to compute a popular matching in the HRLQ problem. As mentioned earlier, our algorithm is a variant of the algorithms in [18]. The algorithm to find a maximal envy-free matching is given in Section 2.2, and its output is a superset of the envy-free matching computed by Yokoi's algorithm [20].

2.1 Algorithm for Popular matching in HRLQ

In this section, we present an algorithm (Algorithm 1) that outputs a feasible matching M in a given HRLQ instance G, such that M is popular amongst all the feasible matchings in G. We assume that G admits a feasible matching, which can be checked in polynomial time [14].

Algorithm 1 is a modification of the standard hospital-proposing Gale and Shapley algorithm [12] and can be viewed as a two-phase algorithm. The first phase simply computes a stable matching M_s in the input instance ignoring lower quotas. If M_s satisfies the lower quotas of all the hospitals, it just outputs M_s. Otherwise hospitals that are *deficient* in M_s propose with *increased priority.* A hospital h is deficient w.r.t. a matching M if $|M(h)| < q^-(h)$. We implement the increased priority by assigning a *level* to each hospital, so that higher level corresponds to higher priority. A resident always prefers a higher level hospital in its preference list to a lower level hospital, irrespective of their relative positions in its preference list. We show that at most $|\mathcal{R}|$ levels suffice to output a feasible matching in the instance, if one exists. We give a detailed description below.

Algorithm 1 Popular matching in HRLQ

```
1: Input : $G = (\mathcal{R} \cup \mathcal{H}, E)$
2: set $M = \emptyset$; $Q = \emptyset$;
3: for each $h \in \mathcal{H}$ do set level($h$) = 0; add $h$ to $Q$;
4: for each $r \in \mathcal{R}$ do set level($r$) = -1;
5: while $Q$ is not empty do
6:     let $h$ = head($Q$);
7:     let $\mathcal{S}_h$ = residents to whom $h$ has not yet proposed with level($h$);
8:     if $\mathcal{S}_h \neq \emptyset$ then
9:         let $r$ be the most preferred resident in $\mathcal{S}_h$
10:        if $r$ is matched in $M$ (to say $h'$) then
11:            if pref($r, h, h'$) == 0 then
12:                add $h$ to $Q$; goto Step 5;
13:            else
14:                $M = M \setminus \{(r, h')\}$;
15:                add $h'$ to $Q$ if $h'$ is not present in $Q$;
16:        $M = M \cup \{(r, h)\}$; level($r$) = level($h$);
17:        if level($h$) == 0 and $|M(h)| < q^+(h)$ then
18:            add $h$ to $Q$;
19:        else if $h \in \mathcal{H}_{lq}$ and $|M(h)| < q^-(h)$ then
20:            add $h$ to $Q$;
21:    else if $h \in \mathcal{H}_{lq}$ and level($h$) ¡ $|\mathcal{R}|$ then
22:        level($h$) = level($h$)+1; add $h$ to $Q$;
23:        // $h$ starts proposing from the beginning of the preference list.
```

Let $G = (\mathcal{R} \cup \mathcal{H}, E)$ be the given HRLQ instance. Let $\mathcal{H}_{lq}$ denote the set of hospitals which have non-zero lower quota – we call $h \in \mathcal{H}_{lq}$ a lower-quota hospital; we call $h \notin \mathcal{H}_{lq}$ a non-lower quota hospital.

Algorithm 1 begins by initializing the matching M and a queue Q of hospitals to be empty (Step 2). As described above, level of a hospital denotes its current priority, and initially all the hospitals have level 0. All hospitals are added to Q (Step 3). We also assign a level to each resident r, which stores the level of the hospital $M(r)$ at the time when r gets matched to $M(r)$. Initially, all residents are assigned level -1 (Step 4). The main while loop of the algorithm (Step 5) executes as long as Q is non-empty. Steps 6–16 are similar to the execution of the hospital-proposing Gale-Shapley algorithm [13]. When a matched resident r gets a proposal from a hospital h, r compares its current match h' with h using **pref**(r, h, h'). We define **pref** $(r, h, h') = 1$ if (i) level$(r) <$ level(h), which means h proposes to r with a higher priority than h' did, or (ii) level$(r) =$ level(h) and h has a higher position than h' in the preference list of r. We define **pref** $(r, h, h') = 0$ otherwise. Thus matched resident r accepts the proposal of h and rejects h' if and only if **pref** $(r, h, h') = 1$. In case the edge (r, h) is added to M (Step 16), the algorithm also sets the level of r equal to the level(h).

When a hospital h finishes proposing all the residents on its preference list with level$(h) = i$ and still $|M(h)| < q^-(h)$, level(h) is incremented by 1 and h is added back to Q. This is done in Step 21. Now h restarts proposing all the residents in its preference list with the new value of level(h).

Running time and correctness: To see that the algorithm terminates we observe that every non-lower quota hospital proposes to residents in its preference list at most once. Every lower-quota hospital proposes to residents on its preference list at most $|\mathcal{R}|$ times. Thus the running time of our algorithm is $O((|\mathcal{R}| + |\mathcal{H}| + |E|) \cdot |\mathcal{R}|)$. To see the correctness note that when G admits a feasible stable matching our algorithm degenerates to the standard Gale-Shapley hospital proposing algorithm. As proved in [19], a stable matching is popular amongst the set of feasible matchings. In case G admits a feasible matching but no feasible stable matching, techniques as in [18] can be employed to show that the output is popular amongst the set of feasible matchings.

Theorem 1. *In an* HRLQ *instance G, Algorithm 1 outputs a matching that is feasible and popular amongst the set of feasible matchings. If G admits a feasible and stable matching, then Algorithm 1 outputs a stable matching.*

2.2 Algorithm for Maximal Envy-free Matching

Yokoi [20] has given an algorithm to compute an envy-free matching in an HRLQ instance $G = (\mathcal{R} \cup \mathcal{H}, E)$. Yokoi's algorithm works in the following steps:

1. Set the upper quota of each hospital to its lower quota.
2. Set the lower quota of each hospital to 0, call this modified instance G_1.
3. Find a stable matching M_1 in the modified instance.
4. Return M_1, if every hospital gets matched to as many residents as its upper quota in G_1, otherwise declare that there is no envy-free matching in G.

It can be seen that Yokoi's algorithm, as stated above, does not return a *maximal envy-free* matching. In particular, any hospital with lower-quota 0 does not get matched to any resident in Yokoi's algorithm. Hence we propose a simple extension of Yokoi's algorithm to compute a maximal envy-free matching, which contains the matching output by Yokoi's algorithm. We call a matching M in G maximal envy-free if, for any edge $e = (r, h) \in E \setminus M$, $M \cup \{e\}$ is not envy-free. The following definition with respect to Yokoi's output matching M_1 is useful for our algorithm. Our algorithm is described as Algorithm 2.

Definition 4. *Let h be a hospital that is under-subscribed in G with respect to M_1, that is $|M_1(h)| < q^+(h)$. A* threshold resident *r_h for h, if one exists, is the most preferred resident in the preference list of h such that (i) r_h is matched in M_1, to say h', and (ii) r_h prefers h over h'. If no such resident exists, we assume a unique dummy resident r_h at the end of h's preference list to be the threshold resident for h.*

Algorithm 2 Maximal envy-free matching in HRLQ

```
1: Input : $G = (\mathcal{R} \cup \mathcal{H}, E)$
2: Compute a matching $M_1$ by Yokoi's algorithm.
3: if Yokoi's algorithm declares "no envy-free matching" then
4:     Return $M = \emptyset$.
5: Let $\mathcal{R}'$ be the set of residents unmatched in $M_1$.
6: Let $\mathcal{H}'$ be the set of hospitals such that $|M_1(h)| < q^+(h)$ in $G$.
7: Let $G' = (\mathcal{R}' \cup \mathcal{H}', E')$ be an induced subgraph of $G$, where $E' = \{(r, h) \mid r \in \mathcal{R}', h \in$
   $\mathcal{H}', h$ prefers $r$ over its threshold resident $r_h\}$. Set $q^+(h)$ in $G'$ as $q^+(h) - q^-(h)$ in $G$.
8: Each $h$ has the same relative ordering on its neighbors in $G'$ as in $G$.
9: $M_2$ = stable matching in $G' = (\mathcal{R}' \cup \mathcal{H}', E')$.
10: Return $M = M_1 \cup M_2$.
```

Below we prove that the output of Algorithm 2 is a maximal envy-free matching.

Theorem 2. *If G admits an envy-free matching, then Algorithm 2 outputs M which is maximal envy-free in G.*

Proof. Since G admits an envy-free matching, M_1 output by Yokoi's algorithm is non-empty. We prove that M is an envy-free feasible matching, and for each edge $e \in E \setminus M$, $M \cup \{e\}$ is not an envy-free matching.
M **is envy-free:** Assume, for the sake of contradiction, that a resident r' has a justified envy towards a resident r with respect to M. Thus r' prefers $h = M(r)$ over $h' = M(r')$, and h prefers r' over r. The edge $(r, h) \in M$ and hence either $(r, h) \in M_1$ or $(r, h) \in M_2$. Suppose $(r, h) \in M_1$. Recall that M_1 is stable in the instance G_1 used in Yokoi's algorithm. In this case, the edge (r', h) blocks M_1 in G_1 a contradiction to the stability of M_1 in G_1.

Thus, if possible, let $(r, h) \in M_2$, and hence $(r, h) \in E'$. If r' is unmatched in M_1, then $(r', h) \in E'$ and (r', h) blocks M_2 in G', contradicting the stability of M_2 in G'. If r' is

matched in M_1 then $r' \notin \mathcal{R}'$ and hence $(r', h) \notin E'$. In this case, the threshold resident r_h of h is either same as r' or is a resident whom h prefers over r'. Since h prefers r' over r, h prefers r_h over r. Therefore (r, h) can not be in E' by construction, and hence $(r, h) \notin M_2$. This proves that M is envy-free.
M **is maximal envy-free:** We now prove that, for any $e = (r, h) \notin M$, $M \cup \{e\}$ is not envy-free. Let $e = (r, h), h \in \mathcal{H}, r \in \mathcal{R}$. Clearly, h must be *under-subscribed in* M i.e. $|M(h)| < q^+(h)$, and r must be unmatched in M, otherwise $M \cup \{e\}$ is not a valid matching in G. Let r_h be the threshold resident of h with respect to M_1. Since r is unmatched in M and hence in M_1, $r \in \mathcal{R}'$. (i) If h prefers r over r_h, then $(r, h) \in E'$ blocks M_2 in G', which contradicts the stability of M_2 in G'. (ii) If h prefers r_h over r, then adding the edge (r, h) to M makes r_h have a justified envy towards r. Thus, $M \cup \{(r, h)\}$ is not envy-free.

3 Experimental Setup

The experiments were performed on a machine with a single Intel Core i7-4770 CPU running at 3.40GHz, with 32GB of DDR3 RAM at 1600MHz. The OS was Linux (Kubuntu 16.04.2, 64 bit) running kernel 4.4.0-59. The code [5] is developed in C++ and was compiled using the clang-3.8 compiler with -O3 optimization level. To evaluate the performance of our algorithms, we developed data generators which model preferences of the participants in real-world instances. We use a limited number of publicly available data-sets as well as real-world data-sets from elective allocation at IIT-Madras. All the data-sets generated and used by us are available at [1].

3.1 Data generation models and available data-sets

There are a variety of parameters and all of them could be varied to generate synthetic data-sets. We focus on four prominent parameters – (*i*) the number of residents $|\mathcal{R}|$, (*ii*) the number of hospitals $|\mathcal{H}|$, (*iii*) the length of the preference list of each resident k, (*iv*) capacity of every hospital *cap* (by default $cap = |\mathcal{R}|/|\mathcal{H}|$). In the synthetic data-sets, all hospitals have uniform capacity and all residents have the same length of preference list. We use the following three models of data generation, and data-sets publicly available from [2].

1. **Master**: Here, we model the real-world scenario that there are some hospitals which are in high demand among residents and hence have a larger chance of appearing in the preference list of a resident. Hospitals on the other hand, rely on some global criteria to rank residents. We set up a geometric probability distribution with $p = 0.10$ over the hospitals which denotes the probability with which a hospital is chosen for being included in the preference list of a resident. Each resident samples k hospitals according to the distribution and orders them arbitrarily. We also assume that there exists a master list over the set of residents. For all the neighbours of a hospital, the hospital ranks them according to the master list of residents.
2. **Shuffle**: This model is similar to the first model except that we do not assume a master list on the residents. The residents draw their preference lists as above. Every hospital orders its neighbours uniformly at random. This models the scenario that hospitals may have custom defined ranking criteria. Our **Shuffle** model is closely inspired by the one described by Mahdian and Immorlica [15].
3. **Publicly available HR with couples data-set**: Other than generating data-sets using the three models described above, we used a freely available data-set [2] by Manlove et al. [17] for the HR problem with couples (HRC problem). The instances were only modified with respect to the preference list of residents that participate in a couple, all other aspects of the instance remain the same. Residents participating in a couple can have different copies of a same hospitals on their preference list which are not necessarily contiguous. The preference list is created by only keeping the first unique copy of a hospital in the preference list of a resident.
4. **Elective allocation data from IIT-M**: We use a limited number of real-world data-sets available from the IIT Madras elective allocation. The data-sets are obtained from the SEAT (Student Elective Allocation Tool) which allocates humanities electives and outside department electives for under-graduate students across the institute every semester. The

input consists of a set of students and a set of courses with capacities (upper-quotas). Every student submits a preference ordering over a subset of courses and every course ranks students based on custom ranking criteria. This is exactly the HR problem.

HRLQ **instances:** The above three data-generation models (Master, Shuffle, and Random) are common for generating preferences in HRLQ and HR data-sets. For HRLQ data-sets we additionally need lower quotas. In all our data-sets, around 90% of the hospitals had lower quota at least 1. This is to ensure that the instances are HRLQ instances rather than *nearly* HR instances. Also, the sum of the lower quota of all the hospitals was kept around 50% of the total number of positions available. This ensures that at least half of the residents must be matched to a lower quota hospital. Lastly, we consider only those HRLQ instances which admit a feasible matching but *no stable feasible matching.* This is done by simply discarding instances that admit a feasible stable matching. We discard such instances because if an instance with feasible stable matching, the stable matching is optimal with respect to popularity, envy-freeness and min-BP and min-BR objectives.
Methodology: For reporting our results, we fix a model of generation and the parameters $|\mathcal{R}|, |\mathcal{H}|, k, cap$. For the chosen model and the parameters, we generate 10 data-sets and report arithmetic average for all output parameters on these data-sets.

4 Empirical Evaluation

Here, we present our empirical observations on HRLQ and HR instances. In each case, we define set of parameters on the basis of which we evaluate the quality of our matchings.

4.1 HRLQ instances

In this section we show the performance of Algorithm 1 and Algorithm 2 on data-sets generated using models described earlier. Let G be an instance of the HRLQ problem, and M_s be the stable matching in the instance ignoring lower-quotas. For all our instances M_s is infeasible. Let M_p denote a popular matching output Algorithm 1 in G. Let M_e denote a maximal envy-free matching output by Algorithm 2. Both M_p and M_e are feasible for G and hence unstable.
Parameters of interest: We now define the parameters of the matching that are of interest in the HRLQ problem. For $M \in \{M_p, M_e\}$ we compute the following.

- $S(M)$: size of the matching M in G.
- $BPC(M)$: number of blocking pairs w.r.t. M in G. Since M is not stable in G, this parameter is expected to be positive; however we would like this parameter to be small.
- $BR(M)$: number of residents that participate in at least one blocking pair w.r.t. M.
- $\mathcal{R}_1(M)$: number of residents matched to their rank-1 hospitals in M.

We additionally compute the deficiency of the stable matching M_s. Hamada et al. [14] showed that $Def(M_s, G)$ is a lower bound on the number of blocking pairs and the number of blocking residents in an HRLQ instance. To analyze the goodness of M_p and M_e w.r.t. the Min-BP and Min-BR objectives, we compare the number of blocking pairs and blocking residents with the lower bound of $Def(M_s, G)$.

- $Def(M_s, G)$: This parameter denotes the deficiency of the stable (but not feasible) matching M_s in G. For every hospital h, let $def(M_s, h) = \max\{0, q^-(h) - |M_s(h)|\}$. The deficiency of the instance G is the sum of the deficiencies of all hospitals.

We now describe our observations for the data-sets generated using various models. For a particular model, we vary the parameters $|\mathcal{R}|, |\mathcal{H}|, k, cap$ to generate HRLQ data-sets using the different models. In all our tables a column with the legend $\uparrow$ implies that larger values are better. Analogously, a column with the legend $\downarrow$ implies smaller values are better.

$\|\mathcal{H}\|$	$Def(M_s,G)$	$S(M) \uparrow$		$BPC(M) \downarrow$		$BR(M) \downarrow$		$\mathcal{R}_1(M) \uparrow$	
		M_p	M_e	M_p	M_e	M_p	M_e	M_p	M_e
100	30.80	**885.40**	559.00	**78.50**	2747.00	**34.80**	822.00	**554.10**	174.00
20	23.60	**897.90**	510.66	**67.70**	3067.33	**27.50**	803.66	**570.40**	195.33
10	27.50	**912.80**	535.50	**85.10**	2945.00	**31.10**	770.87	**600.40**	226.87

Table 1. Data generated using the **Master** model. All values are absolute. $|\mathcal{R}| = 1000, k = 5$.

Popular Matchings versus Maximal Envy-free Matchings Here we report the quality of popular matchings and envy-free matchings on the parameters of interest listed above on different models.

Table 1 shows the results for popular matchings (M_p) and maximal envy-free matchings (M_e) on data-sets generated using the **Master** model.

Table 2 shows the results for popular matchings (M_p) and maximal envy-free matchings (M_e) on data-sets generated using the **Shuffle** model.

$\|\mathcal{H}\|$	$Def(M_s,G)$	$S(M) \uparrow$		$BPC(M) \downarrow$		$BR(M) \downarrow$		$\mathcal{R}_1(M) \uparrow$	
		M_p	M_e	M_p	M_e	M_p	M_e	M_p	M_e
100	17.00	**892.70**	–	**27.20**	–	**19.40**	–	**350.30**	–
20	20.60	**915.30**	547.00	**34.40**	2838.20	**23.60**	808.00	**343.90**	185.80
10	35.40	**930.00**	490.33	**57.80**	3388.00	**35.40**	853.00	**309.30**	147.00

Table 2. Data generated using the **Shuffle** model. All values are absolute.$|\mathcal{R}| = 1000, k = 5$.

We observe the following from the above two tables.

- **Guaranteed existence:** As noted earlier, envy-free matchings are not guaranteed to exist in contrast to popular matchings which always exist in HRLQ instances. For instance, in the **Shuffle** model, for $|\mathcal{R}| = 1000, |\mathcal{H}| = 100, k = 5$ (Table 2, Row 1) none of the instances admit an envy-free matching. Thus, for the columns M_e we take an average over the instances that admit an envy-free matching.
- **Size:** It is evident from the tables that in terms of size popular matchings are about 32%–43% larger as compared to envy-free matchings (when they exist). See Column $S(M)$ in Table 1 and Table 2.
- **BPC and BR:** In terms of the blocking pairs and blocking residents, popular matchings beat envy-free matchings by over an order of magnitude. We remark that to ensure envy-freeness, several hospitals may to be left under-subscribed. This explains the unusually large blocking pairs and blocking residents in envy-free matchings. Furthermore, note that $Def(M_s, G)$ is a lower-bound on both the number of blocking pairs and blocking residents. On all instances, for popular matchings the number of blocking pairs (BPC) is at most 3 times the optimal whereas the number of blocking residents (BR) is close to the optimal value.
- **Number of Envy-pairs:** Although we do not report it explicitly, the number of envy pairs in an envy-free matching is trivially zero and for any matching it is upper bounded by the number of blocking pairs. Since the number of blocking pairs is significantly small for popular matchings, we conclude that the number of envy-pairs is also small.
- **Number of residents matched to rank-1 hospitals:** For any matching, a desirable criteria is to match as many participants to their top-choice. Again on this count, we see that popular matchings match about 15%–38% more residents to their top choice hospital (See Column $\mathcal{R}_1(M)$ in the above tables).

4.2 Results on HR instances

To analyze the quality of popular matchings on various data-sets, we generate an HR instance $G = (\mathcal{H} \cup \mathcal{R}, E)$, compute a resident optimal stable matching M_s, a maximum cardinality popular matching M_p and a popular matching among maximum cardinality matchings

M_m. The matchings M_p and M_m are generated by creating reduced instances G_2 and $G_{|\mathcal{R}|}$ respectively, and executing the resident proposing Gale-Shapley algorithm in the respective instance. Theoretically, we need to construct $G_{|\mathcal{R}|}$ for computing M_m, practically we observe that almost always a constant number suffices say constructing G_{10} suffices. We now describe our output parameters.

<u>Parameters of interest:</u> For any matching M, let $\mathcal{R}_1(M)$ denote the number of residents matched to rank-1 hospitals in M. For two matchings M and M', let $\mathcal{V}_{\mathcal{R}}(M, M')$ denote the number of residents that prefer M over M'. For each instance we compute the following:

- $S(M_s)$: size of the stable matching M_s in G.
- For M to be one of M_p or M_m define:
  - Δ : $\frac{|M|-|M_s|}{|M_s|} \times 100$. This denotes the percentage increase in size of M_s when compared to M. When comparing M_s with either M_p or M_m, Δ is guaranteed to be non-negative. The larger this value, the better the matching in terms of size as compared to M_s.
  - Δ_1 : $\frac{\mathcal{R}_1(M)-\mathcal{R}_1(M_s)}{\mathcal{R}_1(M_s)} \times 100$. This denotes the percentage increase in number of rank-1 residents of M_s when compared to M. As discussed in the Introduction, there is no guarantee that this value is non-negative. However, we prefer that the value is as large as possible.
  - $\Delta_{\mathcal{R}}$: $\frac{\mathcal{V}_{\mathcal{R}}(M,M_s)-\mathcal{V}_{\mathcal{R}}(M_s,M)}{|\mathcal{R}|} \times 100$. This denotes the percentage increase in number of resident votes of M when compared to M_s. Similar to Δ_1, there is no apriori guarantee that more residents prefer M over M_s. A positive value for this parameter indicates that M is *more resident popular* as compared to M_s. That is, in an election where only residents vote, a majority of the residents would like to move from M_s to M.
  - $BP(M)$: $\frac{\text{number of blocking pairs in } M}{|E|-|M|} \times 100$. The minimum value for $BP(M)$ can be 0 (for a stable matching) and the maximum value can be 100, due to the choice of the denominator ($|E| - |M|$). Since matchings M_m and M_p are not stable, this parameter is expected to be positive and we consider it as the *price* we pay to get positive values for Δ, Δ_1 and $\Delta_{\mathcal{R}}$.

We now present our results on data-sets generated using different models. In each case we start with a stable marriage instance (with $|\mathcal{R}| = |\mathcal{H}|$) and gradually increase the capacity. As before, a column with the legend $\uparrow$ implies that larger values are better for that column. Analogously, a column with the legend $\downarrow$ implies smaller values are better.

$|\mathcal{R}| = 1000, k = 5$		M_p vs M_s				M_m vs M_s			
$|\mathcal{H}|$	$S(M_s)$	$\Delta\uparrow$	$BP(M_p)\downarrow$	$\Delta_1\uparrow$	$\Delta_{\mathcal{R}}\uparrow$	$\Delta\uparrow$	$BP(M_m)\downarrow$	$\Delta_1\uparrow$	$\Delta_{\mathcal{R}}\uparrow$
1000	757.90	**11.81**	4.66	-3.49	5.25	**12.79**	5.34	-4.02	6.41
100	823.50	**12.93**	8.57	-1.64	7.41	**13.99**	9.96	-2.80	8.58
20	870.70	**11.65**	12.22	0.24	7.32	**12.25**	14.47	-0.36	7.47
10	890.00	**10.68**	16.32	0.76	2.37	**10.80**	16.57	0.73	2.64

Table 3. Data generated using the **Master** model. All values except $S(M_s)$ are percentages.

Master : Table 3 shows our results on data-sets generated using the **Master** model. Here, we see that for all data-sets we get at least 10.5% increase in the size when comparing M_s vs M_p (column 3) and M_s vs M_m (column 7). The negative value of Δ_1 (columns 5 and 9) indicates that we reduce the number of residents matched to their rank-1 hospitals by at most 4.02% in our experiments and we also marginally gain for smaller values of $|\mathcal{H}|$. The parameter $\Delta_{\mathcal{R}}$ is observed to be positive which shows that a majority of residents prefer M_p over M_s (column 6) and also prefer M_m over M_s (column 10). Finally, the value of BP (columns 4 and 8) goes on increasing as we reduce the value of $|\mathcal{H}|$.

Shuffle : The results obtained on data-sets generated using the **Shuffle** model are presented in Table 4. The size gains are at least 6% when comparing M_s with M_p (column 3) and M_m (column 7). Looking at the values of Δ_1 from column 5 and column 9 we see that the number of residents matched to their rank-1 hospitals are almost always more, with up to

18% getting their rank-1 choices. We also observe that $\Delta_{\mathcal{R}}$ is always positive, which implies that majority of the residents prefer M_p and M_m over M_s.

$\|\mathcal{R}\| = 1000, k = 5$		M_p vs M_s				M_m vs M_s			
$\|\mathcal{H}\|$	$S(M_s)$	$\Delta \uparrow$	$BP(M_p) \downarrow$	$\Delta_1 \uparrow$	$\Delta_{\mathcal{R}} \uparrow$	$\Delta \uparrow$	$BP(M_m) \downarrow$	$\Delta_1 \uparrow$	$\Delta_{\mathcal{R}} \uparrow$
1000	776.80	**9.39**	2.33	0.52	4.27	**10.20**	2.80	-0.14	5.39
100	856.00	**8.56**	3.55	8.72	7.80	**9.23**	4.13	9.79	9.38
20	900.80	**7.10**	5.50	13.87	9.86	**7.52**	6.01	15.55	11.37
10	935.40	**6.03**	16.57	17.35	5.77	**6.15**	16.76	18.02	6.32

Table 4. Data generated using the **Shuffle** model. All values except $S(M_s)$ are percentages.

Processed HR couples data-set: Table 5 shows our results on publicly available HR with couples data-set [2]. As seen from the columns with different Δ values, popular matchings perform favourably on all desired parameters on these data-sets. This is similar to our observations on data generated using the **Shuffle** model. We also investigated the relation between data generated using **Shuffle** model and the data used in this experiment and found that the data-sets are similar in their characteristics. This confirms that **Shuffle** is a reasonable model. Our results on these data-sets confirm that popular matchings perform favourably on variants of the **Shuffle** model.

$\|\mathcal{R}\| = 100, k = 3 \dots 5$		M_p vs M_s				M_m vs M_s			
$\|\mathcal{H}\|$	$S(M_s)$	$\Delta \uparrow$	$BP(M_p) \downarrow$	$\Delta_1 \uparrow$	$\Delta_{\mathcal{R}} \uparrow$	$\Delta \uparrow$	$BP(M_m) \downarrow$	$\Delta_1 \uparrow$	$\Delta_{\mathcal{R}} \uparrow$
90	84.67	**10.10**	6.26	2.79	4.62	**11.81**	8.55	1.20	7.16
50	87.19	**9.61**	8.25	4.53	4.99	**11.01**	10.77	3.12	6.47
20	91.35	**7.92**	13.57	9.30	4.28	**8.41**	14.93	8.22	3.86
10	93.53	**6.61**	19.94	5.34	-1.86	**6.73**	20.43	4.99	-2.00

Table 5. Processed data-sets from [2]. All values except $S(M_s)$ are percentages.

Real world data-sets from IIT-M: Table 6 shows our results on data-sets obtained from the IIT-M elective allocation (the three rows in the table correspond to the Aug–Nov 2016, Jan–May 2017 and Aug–Nov 2017 humanities elective allocation data respectively).

$\|\mathcal{R}\|, \|\mathcal{H}\|, m$, avg. pref. length					M_p vs M_s				M_m vs M_s			
$\|\mathcal{R}\|$	$\|\mathcal{H}\|$	$\|E\|$	*apl*	$S(M_s)$	$\Delta \uparrow$	$BP(M_p) \downarrow$	$\Delta_1 \uparrow$	$\Delta_{\mathcal{R}} \uparrow$	$\Delta \uparrow$	$BP(M_m) \downarrow$	$\Delta_1 \uparrow$	$\Delta_{\mathcal{R}} \uparrow$
483	18	5313	11.00	481	**0.41**	1.32	0	-0.20	**0.41**	1.32	0	-0.20
729	16	4534	6.21	675	**8.00**	31.98	7.39	-5.48	**8.00**	31.98	7.39	-5.48
655	14	2689	4.10	487	**18.27**	16.42	14.97	7.17	**23.81**	29.91	24.06	5.49

Table 6. Real data-sets from IIT-M elective allocation. All values except $S(M_s)$ are percentages.

For each data-set we list the number of students ($|\mathcal{R}|$), the number of courses ($|\mathcal{H}|$), the sum of total preferences (m) and the average preference list over the set of students (*apl*). On an average, each course has a capacity of 50. For each course a custom ranking criteria is used to rank students who have expressed preference in the course. As seen in Table 6, for the Jan–May 2017 (row 2) and Aug–Nov 2017 (row 3) data-sets, popular matchings perform very favourably as compared to stable matching.

References

1. Data sets. `http://www.cse.iitm.ac.in/~meghana/projects/datasets/popular.zip`
2. HR with couples data-set. http://researchdata.gla.ac.uk/303/
3. National Residency Matching Program. www.nrmp.org
4. Scottish Foundation Association Scheme. www.matching-in-practice.eu/the-scottish-foundation-allocation-scheme-sfas
5. Source code repository. https://github.com/rawatamit/GraphMatching
6. Abraham, D.J., Irving, R.W., Kavitha, T., Mehlhorn, K.: Popular Matchings. SIAM Journal on Computing 37(4), 1030–1045 (2007)
7. Biró, P., David F. Manlove, Mittal, S.: Size versus stability in the marriage problem. Theoretical Computer Science 411(16), 1828–1841 (2010)
8. Biró, P., Irving, R.W., Manlove, D.: Popular matchings in the marriage and roommates problems. In: 7th International Conference on Algorithms and Complexity CIAC. pp. 97–108 (2010)
9. Brandl, F., Kavitha, T.: Popular matchings with multiple partners. In: Proceedings of 37th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science. pp. 19:1–19:15 (2017)
10. Cseh, Á.: Trends in computational social choice. pp. 105 – 122. COST (European Cooperation in Science and Technology) (2017)
11. Cseh, Á., Kavitha, T.: Popular Edges and Dominant Matchings. In: Proceedings of the Eighteenth Conference on Integer Programming and Combinatorial Optimization. pp. 138–151 (2016)
12. Gale, D., Shapley, L.: College Admissions and the Stability of Marriage. American Mathematical Monthly 69, 9–14 (1962)
13. Gusfield, D., Irving, R.W.: The Stable Marriage Problem: Structure and Algorithms. MIT Press (1989)
14. Hamada, K., Iwama, K., Miyazaki, S.: The Hospitals/Residents Problem with Lower Quotas. Algorithmica 74(1), 440–465 (2016)
15. Immorlica, N., Mahdian, M.: Marriage, honesty, and stability. In: Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 53–62 (2005)
16. Kavitha, T.: A Size-Popularity Tradeoff in the Stable Marriage Problem. SIAM Journal on Computing 43(1), 52–71 (2014)
17. Manlove, D.F., McBride, I., Trimble, J.: "Almost-stable" matchings in the Hospitals / Residents problem with Couples. Constraints 22(1), 50–72 (2017)
18. Nasre, M., Nimbhorkar, P.: Popular matchings with lower quotas. In: Proceedings of 37th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science. pp. 44:1–44:15 (2017)
19. Nasre, M., Rawat, A.: Popularity in the generalized hospital residents setting. In: Proceedings of the 12th International Computer Science Symposium in Russia. pp. 245–259 (2017)
20. Yokoi, Y.: Envy-free matchings with lower quotas. In: Proceedings of the 28th International Symposium on Algorithms and Computation, ISAAC. pp. 67:1–67:12 (2017)

5 Example Instances

5.1 Instance where a stable matching is favourable over a maximum cardinality popular matching

$\mathbf{r_1} : h_1$	$[\mathbf{0},\mathbf{1}]$	$\mathbf{h_1} : r_3\ r_1$
$\mathbf{r_2} : h_2\ h_4\ h_3$	$[\mathbf{0},\mathbf{1}]$	$\mathbf{h_2} : r_3\ r_2$
$\mathbf{r_3} : h_1\ h_2$	$[\mathbf{0},\mathbf{1}]$	$\mathbf{h_3} : r_2$
$\mathbf{r_4} : h_4$	$[\mathbf{0},\mathbf{1}]$	$\mathbf{h_4} : r_4\ r_2$

Fig. 2. Example instance in which stable matching M_s matches more residents to their rank-1 hospitals than the maximum cardinality popular matching M_p.

Here, we present an HR instance $G = (\mathcal{R} \cup \mathcal{H}, E)$ (in fact a stable marriage instance) which shows that (i) the stable matching M_s in G matches more residents to their rank-1 hospitals than the maximum cardinality popular matching M_m in G and (ii) more number of residents prefer M_s over M_m. This example shows that theoretically there are no guarantees on these parameters for a popular matching.

Let $\mathcal{R} = \{r_1, \ldots, r_4\}$ and $\mathcal{H} = \{h_1, \ldots, h_4\}$ with the preferences of the residents and hospitals as given in Figure 2. All hospitals have an upper quota of 1. The instance admits a unique stable matching $M_s = \{(r_2, h_2), (r_3, h_1), (r_4, h_4)\}$ whereas the maximum cardinality popular matching is $M_p = \{(r_1, h_1), (r_2, h_3), (r_3, h_2), (r_4, h_4)\}$. The stable matching M_s matches three residents to their rank-1 hospitals whereas M_p matches exactly one resident to its rank-1 hospital. Similarly two residents prefer M_s over M_p and exactly one resident prefers M_p over M_s.

5.2 Instance where max. cardinality popular matching is favourable over a stable matching

Consider another instance where $\mathcal{R} = \{r_1, \ldots, r_5\}$ and $\mathcal{H} = \{h_1, \ldots, h_5\}$. All hospitals have a lower quota of 0 and upper quota of 1. The preferences of the residents and the hospitals are as given in 3. The matching $M_s = \{(r_1, h_4), (r_3, h_1), (r_4, h_5), (r_5, h_3)\}$ is stable in the instance whereas the matching $M_p = \{(r_1, h_4), (r_2, h_5), (r_3, h_1), (r_4, h_3), (r_5, h_2)\}$ is a maximum cardinality popular matching in the instance. The matching M_p has three residents matched to their rank-1 hospitals as opposed to M_s which has only two residents matched to their rank-1 hospitals. Furthermore, it is easy to see that more residents prefer M_p over M_s than the other way.

$\mathbf{r_1} : h_5\ h_4$	$[\mathbf{0},\mathbf{1}]$	$\mathbf{h_1} : r_3\ r_5$
$\mathbf{r_2} : h_5\ h_3$	$[\mathbf{0},\mathbf{1}]$	$\mathbf{h_2} : r_5$
$\mathbf{r_3} : h_1$	$[\mathbf{0},\mathbf{1}]$	$\mathbf{h_3} : r_5\ r_2\ r_4$
$\mathbf{r_4} : h_3\ h_5$	$[\mathbf{0},\mathbf{1}]$	$\mathbf{h_4} : r_1$
$\mathbf{r_5} : h_3\ h_2\ h_1$	$[\mathbf{0},\mathbf{1}]$	$\mathbf{h_5} : r_4\ r_1\ r_2$

Fig. 3. Example instance in which max. cardinality popular matching M_p matches more residents to their rank-1 hospitals than the stable matching M_s. More residents prefer M_p over M_s that the other way.