Finding Large Stable Matchings

Robert W. Irving* and David F. Manlove*

Department of Computing Science, University of Glasgow, Glasgow G12 8QQ, UK
Email: {rwi,davidm}@dcs.gla.ac.uk

Abstract

When ties and incomplete preference lists are permitted in the Stable Marriage and Hospitals/Residents problems, stable matchings can have different sizes. The problem of finding a maximum cardinality stable matching in this context is known to be NP-hard, even under very severe restrictions on the number, size and position of ties. In this paper, we present two new heuristics for finding large stable matchings in variants of these problems in which ties are on one side only. We describe an empirical study involving these heuristics and the best existing approximation algorithm for this problem. Our results indicate that all three of these algorithms perform significantly better than naive tie-breaking algorithms when applied to real-world and randomly-generated data sets, and that one of the new heuristics fares slightly better than the other algorithms in most cases. This study, and these particular problem variants, are motivated by important applications in large scale centralized matching schemes.

1 Introduction

Many large-scale centralized matching schemes that allocate applicants to institutions employ variants of the classical Gale-Shapley algorithm [7] to form stable matchings, taking into account preferences expressed by all participants. Possibly the best known such scheme is the National Resident Matching Program (NRMP) in the US [26], which has operated continuously since 1952, and currently matches annually some thirty thousand medical graduates (or residents) to their first hospital posts. Other similar schemes exist for medical graduates in, amongst other places, Canada [25] and Scotland [27], and in a variety of other contexts and countries. Among those that are documented in the literature are school placement in Boston [2, 24] and New York [1], university faculty recruitment in France [3] and university admission in Spain [22] and Hungary [4]. Anecdotal evidence suggests that there are many other such schemes world-wide. It has been convincingly argued and demonstrated, for example in [23], that stability is the key property that contributes to the success and durability of a centralized matching scheme, and therefore that algorithms that produce stable matchings are crucial in such schemes.

Ties in the preference lists

In practice, because the Gale-Shapley algorithm assumes strict preferences, almost all such matching schemes either require all participants to rank their choices in strict order of preference, or use some form of randomisation to break any ties in the preference lists. This may result in arbitrary decisions, either on the part of participants or scheme administrators, who produce a strictly ordered list by discriminating unnecessarily or

*Supported by EPSRC grant EP/E011993/1.

artificially between applicants. It is not difficult to envisage the magnitude of the task faced by a popular hospital in producing a genuine strictly ordered preference list containing possibly hundreds of applicants. Moreover, in some schemes, applicants are ranked on a relatively coarse-grained scale, and this inevitably produces ties in individual preference lists, which then have to be broken in some more or less arbitrary way.

Formal problem description

At the heart of the centralized matching schemes described above are efficient algorithms that essentially solve the *Hospitals-Residents problem* (HR) or the *College Admissions problem*; we adopt the former terminology in view of the pre-eminence of applications in the medical sphere, and of our own involvement in such a real-world matching scheme. The well-known *Stable Marriage problem* [7] is a special case of HR. In view of practical applications where ties may arise in preference lists, as detailed above, we will define formally a generalisation of HR called the *Hospitals / Residents problem with Ties* (HRT).

An instance of HRT comprises a set of n residents $r_1, \ldots, r_n$ and a set of m hospitals $h_1, \ldots, h_m$, each hospital h_j having a *capacity* $c_j \in \mathbb{Z}^+$, indicating its number of available *posts*. Each resident has a *preference list* consisting of a subset of the hospitals, his or her *acceptable assignments*, listed in order of preference. In an arbitrary HRT instance, a resident's preference list may contain one or more ties, each consisting of two or more hospitals of equal preference. However our main focus in this paper will be on the case in which residents' preferences are strict, since, in practice, these preference lists are typically short, and individuals can generally discriminate among their preferred career options. Each hospital has a preference list containing their applicants, namely those residents who have ranked that hospital, but with ties permitted. If hospital h_j precedes hospital h_k on resident r_i's list then r_i is said to *prefer* h_j to h_k, and similarly for residents on a hospital's list.

A *matching* in I is a set M of resident-hospital pairs (r_i, h_j), where h_j is an acceptable assignment for r_i, with each resident appearing in at most one pair of M and each hospital in a number of pairs that is bounded by its capacity. If M is a matching and $(r_i, h_j) \in M$ we write $M(r_i) = h_j$, and we say that r_i is *matched* in M to h_j. Also for any hospital h_j we let $M(h_j) = \{r_i : (r_i, h_j) \in M\}$ (so that $M(h_j)$ is a set). A pair (r_i, h_j) is a *blocking pair* for M, or *blocks* M, if h_j is an acceptable assignment for r_i, r_i is either unmatched in M or prefers h_j to $M(r_i)$, and simultaneously either h_j is undersubscribed, i.e., $|M(h_j)| < c_j$, or prefers r_i to at least one member of $M(h_j)$. A matching for which there is no blocking pair is said to be *stable*. (This notion of stability is also referred to as *weak* stability in contexts where other kinds of stability are discussed [12].)

The restriction of HRT in which each hospital has a capacity of 1 is known as the *Stable Marriage problem with Ties and Incomplete lists* (SMTI); in this setting residents and hospitals may be equated with men and women, and matchings are one-to-one. SMTI is an extension of the classical Stable Marriage problem (SM) introduced by Gale and Shapley [7], in which all preference lists are complete and strict. Gale and Shapley proved that, for every instance of SM, there is at least one stable matching, and they described an $O(n^2)$ time algorithm to find such a matching, where n is the number of men/women; this has come to be known as the Gale-Shapley algorithm. This algorithm is easily extended to the case of incomplete lists and to the Hospitals-Residents problem (without ties), and it can be implemented to run in $O(a)$ time in this case, where a is the sum of the lengths of all of the preference lists – see [9] for details.

Why not employ random tie-breaking?

As mentioned above, in the context of centralized matching schemes where preference lists involve ties, some element of tie-breaking is necessary in order that the Gale-Shapley algorithm can be employed. Arbitrary or random tie-breaking does not, on the surface, seem controversial, and a genuinely random tie-breaking mechanism might well be accepted as a fair way of proceeding in such circumstances. However, the ways in which ties are broken can affect not only the precise details, but, crucially, the *size*, of the matching produced by the Gale-Shapley algorithm [20]. In almost all situations, a larger stable matching is preferable to a smaller one, since unmatched residents will typically be disappointed, and possibly disillusioned, at being unmatched, and will have to enter some secondary process that allocates them to unfilled places, such as the so-called 'scramble' that follows the NRMP match [26]. Indeed, there can be more serious implications; for example, it is known that, in Scotland, there is a genuine risk that residents who are not matched in the first round will seek positions elsewhere, in England or further afield, and hence may be lost to the Scottish healthcare system.

Fundamental results

If all preference lists are strict, then it is known that, for a given problem instance, there may be many stable matchings. Indeed, in the HR context, the Gale-Shapley algorithm may be applied from either the residents' side (the *resident-oriented* version of the algorithm) or the hospitals' side (the *hospital-oriented* version), and in general these two applications will produce different stable matchings. The resident-oriented algorithm constructs the *resident-optimal* stable matching; in this matching, every resident has the best assignment that she can have in any stable matching. On the other hand, the hospital-oriented version produces the *hospital-optimal* stable matching, in which each hospital has, in a precise sense, the best set of residents that it can have in any stable matching. However, what is best possible for the residents turns out to be worst possible for the hospitals, and vice versa – see [9] for a fuller discussion of these optimality issues. Exceptionally, the resident-optimal and hospital-optimal stable matchings may coincide, in which case this is the unique stable matching, but in general there may be other stable matchings – possibly exponentially many – between these two extremes [13]. However, for a given instance of HR, all stable matchings have the same size, match exactly the same set of residents, and fill exactly the same number of posts at each hospital [8, 23].

By contrast, the situation for HRT (and SMTI), is dramatically different. Again, at least one stable matching exists for every instance, and can be found in $O(a)$ time by breaking all ties in an arbitrary way to give an instance of HR, and applying the Gale-Shapley algorithm to that instance. However, the ways in which ties are broken can significantly affect the size of the stable matching found, and in the most extreme case, there may be two stable matchings M and M' with $|M| = 2|M'|$ [20]. Furthermore, the problem of finding a stable matching of maximum cardinality (henceforth a *maximum stable matching*) for an instance of HRT – problem MAX-HRT – is NP-hard [20, 15], even under severe restrictions. For example, NP-hardness holds even if each hospital has capacity 1, each resident's list is strictly ordered, and each hospital's list is either strictly ordered or is a tie of length 2 [20]. Also, MAX-HRT is NP-hard even if each hospital has capacity 1, each preference list is of length at most 3, and each resident's list is strictly ordered [15]. Given that NP-hardness for MAX-HRT holds in each of these two cases when every hospital has capacity 1, it follows that MAX-SMTI, the problem of finding a maximum stable matching given an SMTI instance, is also NP-hard for the same restrictions involving the location and length of the ties, and the preference list lengths.

Approximation algorithms and heuristics

In this study, because of its relevance in practical applications, we focus on the variant of HRT in which all residents' preference lists are strict, and hospitals' lists may contain arbitrary ties. We refer to this restricted version of HRT as *HR with one-sided ties* (HROST). The correspondingly restricted version of SMTI, where all men's lists are strict and women's lists may contain arbitrary ties, is referred to as *SM with one-sided ties and incomplete lists* (SMOSTI). We use the terms MAX-HROST and MAX-SMOSTI for the problems of finding a maximum stable matching, given an instance of HROST and SMOSTI respectively. The remarks in the previous subsection indicate that each of these problems is NP-hard.

The likely intractability of finding maximum stable matchings in these situations leads to interesting questions regarding approximation algorithms and heuristics. The challenge in solving instances of these problems can be viewed as that of finding an appropriate way of breaking all of the ties in the preference lists. This follows from the observation that there must be some way of breaking the ties, followed by application of the classical Gale-Shapley algorithm, that will yield a maximum stable matching.

Given an instance of HRT, any two stable matchings differ in size by at most a factor of 2 [20], and hence it trivially follows that MAX-HRT is approximable within 2 (simply by breaking ties arbitrarily and applying the Gale-Shapley algorithm). A number of improved approximation algorithms for versions of MAX-SMTI were proposed prior to 2008 [16, 11, 17, 10, 18] and are surveyed in more detail in [14]. In many cases these approximation algorithms can also be applied to instances of MAX-HRT and will yield the same performance guarantee (see [14] for more details). The best of these [18] achieves a performance guarantee of 15/8 for arbitrary instances of MAX-SMTI and MAX-HRT.

Considering the MAX-HROST case in particular, Irving and Manlove [14] gave a 5/3 approximation algorithm for the restricted version of the problem in which each tie can only occur at the end of some preference list. Recently, Király [19] described an approximation algorithm for MAX-HROST with a performance guarantee of 3/2. He also gave an approximation algorithm for the general MAX-HRT problem with a performance guarantee of 5/3, which was subsequently improved by McDermid, who described an approximation algorithm with a performance guarantee of 3/2 for the same problem [21].

However, none of the approximation algorithms mentioned above appears to have been evaluated empirically. Nor has there been any investigation into how the sizes of stable matchings vary, in practice, for instances with different characteristics, and as a consequence there is no understanding of how much may be lost by insisting on strict preferences, or employing naïve tie-breaking strategies, in practical matching schemes.

The contribution of this paper

In this paper, we develop two original heuristics for the MAX-HROST problem (both of which, of course, can also be applied in the SMOSTI context). One of the heuristics can be seen as being based upon the hospital-oriented version of the Gale-Shapley algorithm and the other on the resident-oriented version. Heuristic strategies, based on maximum cardinality bipartite matching and network flow respectively, are used to break any impasse reached because of the presence of ties in the preference lists.

We present an empirical analysis of these two heuristics together with Király's algorithm (the approximation algorithm with by far the strongest worst-case performance guarantee for MAX-HROST) and two simple approaches based on random tie-breaking. One key objective of this analysis is to determine if any one of the three algorithms consistently produces the largest stable matchings. As will be reported fully in Section 5,

our results indicate that, at least with respect to our data sets, the new resident-oriented heuristic out-performs the other algorithms in the vast majority of cases.

We also use this empirical study to accumulate evidence, hitherto entirely absent from the literature, on variations in stable matching sizes for instances of HROST. Versions of the heuristics studied here have been successfully incorporated into the Scottish Foundation Allocation Scheme (SFAS), the centralized matching scheme for allocating graduating medical students to Foundation positions (as junior doctors) in Scottish hospitals [27]. We report on experience with the algorithms as applied to anonymized real-world data arising from previous runs of SFAS as well as with artificially generated instances having a range of values for various problem parameters.

The structure of this paper

The remainder of this paper is structured as follows. In Section 2 we describe a hospital-oriented heuristic, with appropriate motivation for the approach embodied in it. A similar treatment of an alternative resident-oriented heuristic is given in Section 3. Section 4 summarises the recent elegant approximation algorithm for MAX-HROST due to Király [19]. Section 5 describes the empirical studies that we undertook using both real and artificial data, and summarises the outcome of these studies, focusing on a comparison of the effectiveness of the three algorithms as compared to two variants of an approach based on random tie breaking. As a by-product of these comparisons we gain some feel for the variation in the sizes of stable matchings for problem instances with various parameter values, which gives an indication of the significance of the performance of the heuristics.

2 Hospital-oriented heuristic

In what follows, we assume that each resident's preference list is strict, and each hospital's preference list contains arbitrary ties. In this section we present Algorithm HROST-Heuristic-H, a hospital-oriented heuristic for MAX-HROST that is a development of the approximation algorithm of Irving and Manlove [14].

2.1 Procedure `Hospitals-offer`

At the heart of this first heuristic is a procedure that is based on the hospital-oriented version of the Gale-Shapley algorithm. In most matching schemes, it is the resident-oriented version of the algorithm that is used, and our approach need not violate such a policy – a hospital-oriented heuristic is used in determining how tie-breaking should be carried out, but once this is established, the resident-oriented version of the Gale-Shapley algorithm can be applied to the resulting strict preference lists to determine the final matching. (Note that this procedure would not result in any change to the size of the stable matching that would be obtained by using the hospital-oriented version of the Gale-Shapley algorithm throughout, since, as noted in Section 1, all stable matchings for a given HR instance have the same size.)

Each hospital offers posts to residents on its preference list, but stops when it reaches a tie with size greater than the number of posts it still has left to offer. When a resident r_i receives an offer from a hospital h_j, she accepts it, and becomes (provisionally) assigned to that hospital, rejecting any offer that she already holds. In addition, all successors of h_j in r_i's list are deleted, and r_i is deleted from their lists; such pairs play no further part in the algorithm, as they could not be part of a stable matching for the current instance, nor could they form a blocking pair for any matching obtained by breaking ties in the current preference lists and running the Gale-Shapley algorithm. (The current instance

```
while (there is a hospital h_j such that v_j ≥ t_j > 0)
    for (each resident r_i in T_j) {
        if (r_i is already assigned)
            unassign r_i;
        assign r_i to h_j;
        for (each hospital h_k that is a successor of h_j in r_i's list)
            delete the pair (r_i, h_k) from the preference lists;
    }
```

Figure 1: Procedure `Hospitals-offer`

will in general be a *refinement* of the original instance, in which some tie breaking has been carried out, but any matching that is stable for this refined instance will automatically be stable for the original instance.)

Rejections and deletions that take place during this process may allow additional hospitals to offer places further down their preference lists, so the procedure continues until no hospital is in a position to make further offers, in other words, until every hospital has either reached a tie containing a number of residents that exceeds the number of remaining places that it has to offer (which may, of course, be zero in some cases), or has offered a post to every resident remaining on its list.

To give a more formal description of this procedure, we introduce some additional terminology and notation. For a hospital h_j, the *active tie*, denoted by T_j, is the tie, of length 1 or more[1], that immediately follows, in h_j's current preference list, the least preferred resident that is currently assigned to h_j, should such a tie exist. If such a tie does not exist, we define $T_j = \emptyset$. If $T_j \neq \emptyset$, we let t_j denote the length of T_j, otherwise we let $t_j = 0$. It follows that $t_j = 0$ if and only if all residents in h_j's current list are assigned to h_j. Note that t_j is the length of T_j in the *current* preference list; because of deletions, this may be less than the length of that tie in the original preference list. The number of *vacancies* v_j for hospital h_j is equal to the difference between its capacity c_j and the number of residents currently assigned to it. A hospital h_j is *full* or *undersubscribed* at a given time according as $v_j = 0$ or $v_j > 0$ respectively. Hospitals can alternate between being full and undersubscribed as the algorithm proceeds.

A pseudocode description of this procedure, referred to as procedure `Hospitals-offer`, appears in Figure 1. It is well known that the outcome of the classical hospital-oriented Gale-Shapley algorithm is independent of the order in which hospitals offer posts. It is straightforward to show that this independence applies in this context also, whenever procedure `Hospitals-offer` is invoked.

Procedure `Hospitals-offer` will be re-activated repeatedly during the execution of the heuristic, typically after a decision has been made as to how some ties are to be (at least partially) broken. Hence, whenever the procedure is invoked, it is operating on an HROST instance – we call it the *current* instance – that is a refinement of the original instance, and any matching that is stable for the current instance must also be stable for the original instance. (Of course, the converse is not necessarily true; by resolving or partially resolving just one tie, we may render unstable some matchings that are stable for the original instance.)

The following lemma motivates our tie-breaking strategy for Algorithm HROST-Heuristic.

Lemma 2.1. *(i) If a pair is deleted during an execution of procedure Hospitals-offer, then that pair cannot belong to any stable matching for the current instance.*

[1]Here, and henceforth, we make the simplifying assumption that a single element in a given preference list that is not tied with any other element constitutes a "tie" of length 1.

(ii) If a resident r_i becomes assigned at some point during an execution of procedure Hospitals-offer, then r_i is matched in every stable matching for the current instance.

Proof. (i) Suppose, for a contradiction, that (r_i, h_j) is a pair that belongs to a matching M that is stable for the current instance, and that it was the first such pair deleted by the algorithm. The deletion must have taken place because r_i received an offer from a hospital, say h_k, that she prefers to h_j. Now h_k could only have made that offer to r_i because there were at most c_k residents at least as preferable as r_i in its current preference list. So, in M, h_k must either be undersubscribed, or must be assigned a resident lower in its list than r_i – for, by our assumption, none of the previously deleted residents on the list of h_k can be assigned to h_k in any stable matching. Hence (r_i, h_k) blocks M, a contradiction.
(ii) Suppose that resident r_i becomes assigned to hospital h_j at some point during the algorithm's execution, and that there is a stable matching M for the current instance in which r_i is unmatched. Then to avoid (r_i, h_j) being a blocking pair for M, h_j must be matched in M to c_j residents, none of which is inferior to r_i. However, in order for h_j to have offered a post to r_i, at least one of these residents must have been deleted from h_j's list, contradicting (i). □

We refer to the residents who are assigned at a given point during the execution of Algorithm HROST-Heuristic-H as the *X-residents* and the other residents as the *Y-residents*, and we denote these sets by X and Y respectively. Note that, once a resident enters the set X she will never again be in the set Y. So one way of viewing the progress of the algorithm is that it gradually transfers residents from set Y to set X, and the key objective is to transfer as many residents as possible into X.

2.2 Procedure `Match-Y-residents`

Following an execution of procedure `Hospitals-offer`, we have to decide what happens next. We say that a hospital h_j is *open* if it is undersubscribed and not all of the residents in its current preference list are assigned to it. We let Z denote the set of open hospitals. From the terminating condition for procedure `Hospitals-offer`, we deduce that an open hospital h_j is one for which $t_j > v_j > 0$. The difficulty is that we need to make a decision on breaking the active tie of at least one open hospital in order to make progress, and the question is how this should be done.

Given our strategy of seeking to transfer residents from set Y to set X, the obvious thing to do at this point is to promote, if possible, one or more Y-residents from the active tie T_j of each open hospital h_j, so that offers can be made to these Y-residents, thereby converting them to X-residents. In order to maximise the number of such promotions and subsequent offers, we should find a maximum cardinality matching M between Y-residents and open hospitals, where the pair (r_i, h_j) can be in M if r_i is in T_j. Moreover, each Y-resident r_i can appear at most once in M, and each open hospital h_j can appear at most v_j times in M; we say that the *matching capacity* of r_i is 1 and that of h_j is v_j. Such a matching, in which the capacity of some of the participants is greater than 1, is more properly called a *degree-constrained subgraph* [6]. Once such a maximum cardinality degree-constrained subgraph has been identified, the appropriate promotions can be made from the relevant active ties, allowing procedure `Hospitals-offer` to be re-activated.

Procedure `Match-Y-residents` is summarised in Figure 2. Procedures `Hospitals-offer` and `Match-Y-residents` are invoked repeatedly until the graph G constructed in the latter procedure has no edges, in other words, until no Y-resident appears in the active tie of an open hospital. Notice that `Match-Y-residents` returns the matching M, allowing us to detect this terminating condition by virtue of the fact that M is empty.

$G = (Y \cup Z, E)$, where $(r_i, h_j) \in E$ if and only if $(r_i, h_j) \in Y \times Z$ and $r_i \in T_j$;
for (each resident $r_i \in Y$)
    assign r_i to have matching capacity 1;
for (each hospital $h_j \in Z$)
    assign h_j to have matching capacity v_j;
M = a maximum cardinality degree-constrained subgraph in G;
for (each pair $(r_i, h_j) \in M$)
    promote r_i ahead of the tie T_j;
return M;

Figure 2: Procedure `Match-Y-residents`

2.3 The complete heuristic

At this point, in general, we need to make some further decisions on (partial) tie resolution. Further progress can be made only by breaking one or more of the active ties of open hospitals. We cannot promote any more Y-residents from any of these ties, so we simply break all of these ties randomly[2]. This allows a further round of iterations of procedures `Hospitals-offer` and `Match-Y-residents`. This whole process is then repeated until no open hospitals remain, in other words, until every hospital is either full or has assigned to it all of the residents remaining on its preference list. The complete algorithm is summarised in Figure 3.

do {
    do {
        Hospitals-offer;
        M = Match-Y-residents;
    } while ($M \neq \emptyset$);
    for (each $h_j \in Z$)   // *Z is the set of open hospitals*
        break the active tie T_j randomly;
} while ($Z \neq \emptyset$);
return the current assignment;

Figure 3: The overall HROST heuristic, Algorithm HROST-Heuristic-H

Theorem 2.1. *The matching returned by the above heuristic is a stable matching for the original HROST instance I.*

Proof. In reality, Algorithm HROST-heuristic-H merely breaks ties, in very particular ways, during an application of the classical Gale-Shapley algorithm. It is clear that the assignment M returned by the algorithm is a stable matching in the final HR instance, so no pair $(r_i, h_j) \notin M$ that was not deleted can block M in I. What remains is to show that M is stable in the original HROST instance I. To show this, we require to prove that no pair (r_i, h_j) that was deleted could block M in I. Such a pair (r_i, h_j) was deleted because r_i received an offer from some hospital h_k that she prefers to h_j. At that point r_i is transferred to X, and by part (ii) of Lemma 2.1, r_i is matched in M. Since all hospitals inferior to h_k on r_i's list were deleted, it follows that r_i is matched to a hospital at least as good as h_k in M. Hence (r_i, h_j) does not block M in I. □

[2]It might be asked whether there could be an advantage in choosing just one open hospital at random, and breaking just that one active tie before invoking procedure `Hospitals-offer` again. Empirical evidence suggests that there is no consistent benefit in so doing.

2.4 Complexity of Algorithm HROST-Heuristic-H

To derive a bound on the worst-case complexity of Algorithm HROST-Heuristic-H, we first observe that the total work done during all calls to procedure `Hospitals-offer` is $O(a)$, and a similar bound holds for all the work done during random tie-breaking. It remains to consider procedure `Match-Y-residents`. It is immediate that this procedure can be called at most n times, since at least one resident is moved out of set Y after each call. Suppose that the number of edges in the subgraph M on the ith call is p_i. During each call, the construction of the graph G and the execution of the various for loops can be done in $O(a)$ time, and therefore these operations summed over all calls of the procedure require $O(na)$ time. As far as finding the subgraph M is concerned, this can be achieved in $O(p_i a)$ time by a straightforward augmenting path algorithm, or in $O(\sqrt{p_i}a)$ time using Gabow's algorithm [6]. In either case, summing over i, and using the fact that $\sum p_i \leq n$, we obtain a $O(na)$ bound on the total number of operations required to find the subgraph M, summed over all calls of the procedure. Hence the overall worst-case complexity is $O(na)$ (though a tighter analysis may be possible).

3 Resident-oriented heuristic

As mentioned earlier, our second heuristic, Algorithm HROST-Heuristic-R, is based on the resident-oriented version of the Gale-Shapley algorithm. The idea is that each resident r_i applies to the hospitals on her list, in order. Hospital h_j rejects resident r_i only when it holds applications from at least c_j residents that it strictly prefers to r_i, and as a consequence it is possible for more than c_j residents to be simultaneously (but temporarily) assigned to h_j. To describe this algorithm we require some additional terminology.

We define an *allocation* to be an assignment of each resident to at most one hospital so that, if a hospital h_j has a_j assignees and $a_j > c_j$ then there is a positive integer $u_j > a_j - c_j$ such that the least preferred u_j assignees of hospital h_j are all tied in that hospital's list. (So such a hospital h_j needs a tie-breaking strategy to decide which of these u_j residents should be rejected.)

The concept of a blocking pair is defined for an allocation exactly as for a matching, and an allocation is *stable* if it admits no blocking pairs. We shall see that Algorithm HROST-Heuristic-R, in general, finds a sequence of stable allocations, terminating with a stable matching. (A stable matching is, of course, a stable allocation that respects the capacity restrictions of all of the hospitals.)

3.1 Procedure `Residents-apply`

Each resident applies, in turn, to the first hospital on her preference list. (As was the case with hospitals offering posts, the order in which residents apply does not affect the eventual outcome.) Each hospital initially has no assignees. At any stage during the application of the algorithm, a hospital h_j with a_j assignees is said to be *over-subscribed* if $a_j > c_j$, *full* if $a_j = c_j$, and *under-subscribed* if $a_j < c_j$.

When resident r_i applies to hospital h_j she is assigned (at least temporarily) to that hospital, and a_j is incremented. If, as a result, h_j is full or over-subscribed, then all strict successors, in h_j's preference list, of its c^{th} choice assignee[3], are deleted (and h_j is deleted from their lists), where $c = c_j$. If a resident r_k currently assigned to h_j is thereby deleted from h_j's list, then it follows that the first hospital on r_k's (current) list, namely h_j, is also deleted from that list, the assignment of r_k to h_j is broken, and a_j is decremented.

[3]For the purposes of determining h_j's c^{th} choice assignee we may randomly order the members of any tie containing h_j's worst assignees, if necessary.

```
while (some resident r_i is free and has a non-empty list) {
    h_j = first hospital on r_i's list;
    assign r_i to h_j;
    increment a_j;
    if (a_j ≥ c_j) {      // h_j is full or over-subscribed
        r_k = one of h_j's c^th-choice assignees, where c = c_j;
        for each strict successor r_l of r_k in h_j's list {
            if (r_l is assigned to h_j) {
                break the assignment;
                decrement a_j;
            }
            delete the pair (r_l, h_j);
        }
    }
}
```

Figure 4: Procedure `Residents-apply`

Resident r_k will then subsequently apply to the hospital, if any, that is now at the head of her (reduced) list.

This algorithm is formally described in Figure 4. In fact, this procedure will typically be called several times during the execution of the heuristic; prior to the first call, each resident is assigned to be free, each hospital is assigned to be empty, and a_j is set to zero for each hospital h_j.

Note that a hospital h_j can have $a_j > c_j$, but if this is the case then the number u_j of its least preferred assignees must be greater than $a_j - c_j$, so that procedure `Residents-apply` produces an allocation, as defined earlier.

For a hospital that is full or over-subscribed, we refer to the tie containing its u_j least preferred assignee(s) as the *tail* of its current list. The terminology is chosen because any strict successors of the tail must have been deleted from the list during the application of procedure `Residents-apply`.

The overall resident-oriented heuristic consists of a sequence of applications of this procedure, interleaved with certain tie-breaking decisions. As in Section 2, we refer to the refinement of the original problem instance obtained by tie-breaking decisions taken up to a given point in the algorithm's execution as the *current* instance. As before, any matching that is stable for the current instance is automatically stable for the original, but not necessarily vice-versa.

The following lemma, somewhat analogous to Lemma 2.1, motivates our tie-breaking strategy for Algorithm HROST-Heuristic-R.

Lemma 3.1. *(i) If a hospital becomes full during the application of procedure* `Residents-apply` *then it never again becomes under-subscribed.*
(ii) If a pair is deleted during an execution of procedure `Residents-apply`*, then that pair cannot belong to any stable matching for the current instance.*
(iii) Procedure `Residents-apply` *terminates with a stable allocation.*
(iv) If a hospital h_j is assigned to a_j residents at some point during an execution of procedure Residents-apply, then h_j has at least $\min(a_j, c_j)$ assigned residents in every stable matching for the current instance.

Proof. (i) This is immediate from the condition governing the 'unassignment' of residents.
(ii) Suppose, for a contradiction, that the pair (r_i, h_j) was the first pair belonging to a stable matching, say M, that was deleted during an execution of procedure `Residents-apply`. This deletion took place because h_j had at least c_j assignees, say $r_{i_1}, \ldots, r_{i_{c_j}}$, that it prefers

to r_i. At that point, h_j must have been at the head of the list of each of these residents r_{i_j}, so that, by our assumption, none of these residents can be matched in M to a hospital that they prefer to h_j. Hence, since r_i is matched to h_j in M, at least one of the residents r_{i_k} is matched in M to an inferior hospital (or is unmatched), and it follows that (r_{i_k}, h_j) form a blocking pair for M – a contradiction.
(iii) It follows at once from the condition for the deletion of pairs that, for an over-subscribed hospital h_j, the number a_j of residents assigned to h_j, and the number u_j of these that are in h_j's tail, satisfy $u_j > a_j - c_j$, so that we do indeed have an allocation. In this allocation A, every resident is matched to the hospital at the head of her list. Suppose that (r_i, h_j) is a blocking pair for A. Since r_i prefers h_j to her assigned hospital (or is unassigned in A), the pair (r_i, h_j) must have been deleted, so that, h_j must prefer all of the surviving residents in its list to r_i. Furthermore, h_j must have been full when the deletion took place, and therefore by part (i), is also at least full in A with assignees whom it prefers to r_i, a contradiction.
(iv) Suppose that hospital h_j is assigned residents $r_{i_1}, \ldots, r_{i_a}$ at some point during the execution of procedure `Residents-apply`. Then, at that point, each of these residents has h_j at the head of its list, and so, by part (i), none of them can have a better partner than h_j in any stable matching. It follows that, if M is a stable matching, then, to avoid a blocking pair, h_j is either full, or has all of $r_{i_1}, \ldots, r_{i_a}$ as assignees, from which the result follows. □

From Lemma 3.1(iv) we immediately deduce a lower bound on the size of any stable matching, as indicated in the following theorem.

Theorem 3.1. *Suppose that on termination of an execution of procedure Residents-apply, hospital h_j has a_j assignees. Then any stable matching for the current instance has size s satisfying*

$$s \geq \sum \min(a_j, c_j)$$

where the sum is taken over all hospitals h_j.

For a given stable allocation A, we define the *bound* of A, denoted by $b(A)$, as

$$b(A) = \sum \min(a_j, c_j),$$

and the *excess* of A, denoted by $e(A)$, as

$$e(A) = \sum \delta(a_j - c_j),$$

where $\delta(x) = x$ if $x \geq 0$ and $\delta(x) = 0$ otherwise, and each sum is taken over all hospitals h_j. So a stable allocation of excess zero is a stable matching.

3.2 Procedure `Partially-resolve-ties`

On termination of procedure `Residents-apply`, a stable matching can be found as follows (a method essentially equivalent to breaking remaining ties in an arbitrary way). For each over-subscribed hospital h_j, demote from the tail $a_j - c_j$ randomly chosen assignees. Then re-activate procedure `Residents-apply`. Continue in this way until the algorithm terminates with a matching. It is not hard to show that the resulting matching must be stable.

Of course, the arbitrary choices of deletions will, in general, affect the size of the stable matching generated. Algorithm HROST-Heuristic-R takes a more intelligent approach, and attempts to make tie-breaking decisions that increase, as far as possible, the number of

applications to under-subscribed hospitals. These decisions are made on the basis of a maximum flow in a network constructed from the current stable allocation. Once partial tie-breaking, based on this maximum flow, has been carried out, procedure `Residents-apply` is re-activated, leading to an amended stable allocation whose bound is increased over that of the previous allocation by the value of a maximum flow in the network. Successive iterations of procedures `Residents-apply` and `Partially-resolve-ties` are carried out until the network generated has a maximum flow of zero.

Constructing the flow network

From the current preference lists and current stable allocation A, in which each resident is assigned to the hospital at the head of her list, the corresponding flow network N_A is constructed as follows:

- There is a single source node s and a single sink node t.
- There is a node for each hospital h_j
  - for each over-subscribed hospital, node h_j has an incoming edge from the source s of capacity $a_j - c_j$. This represents h_j's excess number of assignees.
  - for each under-subscribed hospital, node h_j has an outgoing edge to the sink t of capacity $c_j - a_j$. This represents the number of unfilled posts at h_j.
- There is a node for each resident r_i who is a tail assignee of a full or over-subscribed hospital h_j, and who has at least one other hospital on her current list. Let the sequence of consecutive entries following h_j on r_i's current list be $h_{r,1}, h_{r,2}, \ldots, h_{r,k}$, where $h_{r,k}$ is the first of these hospitals such that (a) $h_{r,k}$ is under-subscribed, or (b) r_i is not in the tail of $h_{r,k}$, or (c) $h_{r,k}$ is the last hospital in r_i's list.

  The node representing r_i has an incoming edge from h_j of capacity 1, and for each s ($1 \leq s \leq k$), an outgoing edge to $h_{r,s}$ of capacity 1. The intuition here is that we are prepared to switch r_i from h_j to any such $h_{r,s}$, because no blocking pair could result, but not, at least for now, to any hospital lower than $h_{r,k}$ on her list.

In what follows, we refer to the node representing hospital h_j (respectively resident r_i) simply as the node h_j (respectively the node r_i).

Consider a maximum (integral) flow F in network N_A. Relative to this maximum flow, we call a resident r_i *movable* if there is a flow of 1 through the node r_i in N. The flow must reach r_i along an edge from a hospital node h_j and must leave r_i along an edge to another hospital node h_k; we denote this hospital h_k by $F(r_i)$. The flow is used to partially resolve some of the ties in the preference lists by demoting such a movable resident r_i from the tie in the list of any hospital preceding h_k on her list, including that of h_j. By *demoting* r_i from a tie T that contains it, we mean moving r_i to be a non-tied entry immediately following T. Multiple residents demoted from the same tie will appear in arbitrary strict order immediately following that tie. Note that a demotion of resident r_i within the list of hospital h_j can only happen when h_j is either full or over-subscribed, and r_i appears in the tail of h_j.

Procedure `Partially-resolve-ties` is summarised in Figure 5. The procedure returns a boolean value to indicate whether any ties were, in fact, resolved.

Following these preference list demotions, reactivation of procedure `Residents-apply` will lead to the rejection of the 'moved' residents by the hospitals in question, so that these residents apply to, and become assigned to, hospitals that were previously under-subscribed.

```
construct network N_A from the current preference lists and allocation A;
find a maximum flow F in N_A;
if F has value > 0
    for each movable resident r_i relative to F
        demote r_i from the tail of each predecessor of F(r) on r_i's list;
    return true;
else
    return false;
```

Figure 5: Procedure `Partially-resolve-ties`

Theorem 3.2. *If a maximum flow in network N_A has value f then application of procedure* `Partially-resolve-ties` *followed by procedure* `Residents-apply` *leads to a stable allocation with bound $b(A) + f$ and excess $e(A) - f$.*

Proof. After applying the two procedures, it is clear that each movable resident r_i relative to the maximum flow will be assigned to $F(r_i)$ in the new allocation. Every unit of flow passes through a unique resident vertex, and causes a previously oversubscribed hospital to have one fewer assignee and each undersubscribed hospital to have one additional assignee. Furthermore, the capacities on the edges from the source ensure that no previously oversubscribed hospital can become undersubscribed, and the capacities on the edges to the sink ensure that no previously undersubscribed hospital can become oversubscribed. Also, Kirchhoff's law (the fact that the flow into each vertex is equal to the flow out of that vertex) ensures that each full hospital remains full. Hence every unit of flow adds one to the bound, and subtracts one from the excess, of the allocation. The stability of the allocation follows by an argument analogous to that of the proof of Lemma 3.1(iii). □

Using a maximum flow in network N_A to produce a new stable allocation may actually reduce the excess of the allocation to zero, giving a stable matching, and resulting in termination of Algorithm HROST-Heuristic-R.

Otherwise we can repeat the whole process, since it is feasible that the network constructed from the new stable allocation might have a non-zero flow. This may be the case, for example, if an undersubscribed hospital becomes full, so that residents in the tail of that hospital will give rise to vertices that were not previously present in the network. In fact, the process can be iterated until an allocation of excess zero results, or a network with zero flow is obtained.

3.3 The overall algorithm

If successive applications of procedures `Residents-apply` and `Partially-resolve-ties` lead to a network with zero flow and a stable allocation that is not a matching, in other words that has positive excess, we must decide how to proceed. In order to allow the two procedures to make further progress towards a stable matching, we have to make some additional intervention in terms of breaking ties. The simplest possibility is merely to break the tail ties of oversubscribed hospitals randomly, and then restart the whole process. This sequence of operations can be continued until, finally, we have an allocation of zero excess, in other words a stable matching.

The overall algorithm is summarised in Figure 6.

```
while (true) {
    do {
        Residents-apply;
        Partially-resolve-ties;
    } while (flow in network > 0);
    if (current allocation has excess > 0)
        break randomly tail ties of oversubscribed hospitals;
    else
        return current allocation;     // which is a stable matching
}
```

Figure 6: The overall HROST heuristic Algorithm HROST-Heuristic-R

3.4 Complexity of Algorithm HROST-Heuristic-R

For a bound on the worst-case complexity of Algorithm HROST-Heuristic-R, we first observe that the total number of steps summed over all calls to the `Residents-apply` procedure is $O(a)$. Likewise, the total number of steps in all random-tie breaking operations is $O(a)$.

That leaves the `Partially-resolve-ties` procedure. Because the number of edges in the network is always $O(a)$, any call of this procedure that results in a flow of $f > 0$ can be achieved by breadth-first search augmentation in $O(fa)$ time – f augmentations costing $O(a)$ time each. Construction of the graph can also be achieved in $O(a)$ time. Now, the sum of the f's is bounded by n, since each unit of flow results in an increase of 1 in the bound of the current allocation. So summing these contributions over all calls gives a bound of $O(na)$.

Finally, we account for calls of the `Partially-resolve-ties` procedure that yield zero flow. Consider the edges in the network in such a call. There are at most m edges incident from the source and to the sink, and every other edge has a resident vertex as one of its end-points. Of the edges incident to or from a resident vertex, all but one correspond to an entry in a tie in a preference list that will be broken immediately after the flow of zero is returned. So the total number of edges is at most $n + m + x$, where at least x tied entries become untied as a consequence of the zero flow. It follows that the sum of the x's over all such calls of the procedure is bounded by a. Since there cannot be more than a such calls, the total number of steps summed over all of these calls is $O((n + m)a)$, and this is $O(na)$ if we make that natural assumption that $m = O(n)$.

In summary, the contributions of all of these components to the worst-case complexity is $O(na)$ so this is an overall complexity bound for the heuristic.

4 Király's approximation algorithm

Recently, Király [19] described ingenious approximation algorithms for MAX-SMTI and MAX-HRT with improved performance guarantees. The algorithm for the case of one-sided ties, the problems MAX-SMOSTI and MAX-HROST, has a performance guarantee of 3/2. In this section we summarise the algorithm for the HRT case, using our tie-breaking terminology in place of the concept of 'extra scores' favoured by Király.

The algorithm is based on 'proposals' from residents to hospitals. A hospital accepts a proposal if it is undersubscribed or if it is full but strictly prefers the proposer to its least favoured assignee – according to its current preference list, which may be a refinement of the original. In the latter case, the least favoured assignee is rejected. This assignee is chosen at random if two or more least favoured assignees are tied.

```
while (∃ resident r_i: r_i is unmatched and (unpromoted or unexhausted)) {
    if (r_i is exhausted) {
        promote r_i;  // r_i must have been unpromoted
        set r_i to be unexhausted;  // r_i will restart proposals from the start of her list
    }
    h_j = next hospital on r_i's list;  // the next one in line for a proposal
    r_i proposes to h_j;
    if (h_j is undersubscribed)
        assign r_i to h_j;
    else if (h_j prefers r_i to its least favoured assignee r_k) { // according to its current list
        h_j rejects r_k;
        unassign r_k from h_j;
        assign r_i to h_j;
    }
    else
        h_j rejects r_i;  // and r_i remains unassigned
}
return the current matching;  // which is a stable matching
```

Figure 7: Király's approximation algorithm for HROST

If a resident is rejected by all of the hospitals on her list – we say that the resident is *exhausted* – she is then *promoted*; this means that every tie that currently contains her is partially broken by moving her strictly ahead of the other members of the tie. On entering the promoted state, a resident is allowed to re-start her proposal sequence from the beginning of her preference list. Only if and when she has been rejected by all the hospitals in her list, after having been promoted, does she cease to play any further part in the algorithm and is doomed to be unmatched in the final matching. A pseudocode description of Király's algorithm appears in Figure 7.

For full details of the algorithm, and a proof of the 3/2 performance guarantee, see [19]. As far as complexity is concerned, Király's algorithm can be viewed as an extension of the resident-oriented version of the Gale-Shapley algorithm in which each resident may traverse his preference list twice rather than once. With appropriate data structures, book-keeping details such as promotion can be handled with no adverse effect on the complexity, so that, like the Gale-Shapley algorithm, Király's algorithm can be implemented to run in time that is linear in the sum of the lengths of the preference lists.

5 An Empirical Study

The empirical study is based on the limited examples of real-life data that we have – namely, those for the 2006, 2007 and 2008 runs of the SFAS matching scheme – together with a range of examples of artificially generated data with various combinations of parameter values.

The main objective in conducting this study was to ascertain the effectiveness of the two heuristics presented in this paper, together with Király's algorithm, in diverse sets of circumstances, as compared to two algorithms based on random-tie breaking followed by execution of the classical Gale-Shapley algorithm, bearing in mind our focus on the sizes of stable matchings found. A subsidiary aim was to collect evidence on the spread of stable matching sizes as we varied the values of certain problem parameters.

It is clear from the description of the algorithms given in Sections 2, 3 and 4 that random tie-breaking does form a component of all three of the algorithms presented here.

Depending on how ties are broken whenever these steps of the algorithms are invoked, a stable matching of greater or lesser size may result. Obviously, the same observation applies to algorithms based exclusively on random tie-breaking followed by execution of the classical Gale-Shapley algorithm. The question arises as to how to obtain a fair and reasonable comparison between the various algorithms. One possibility would have been to allow each algorithm the same fixed number of iterations, and to report the distribution of stable matching sizes across the iterations, highlighting the maximum, minimum and average sizes. However, this could be construed as favouring the more complex algorithms, in particular Algorithm HROST-Heuristic-R, which in practice executes significantly more slowly than all of the others. It seemed to be a fairer alternative to give each algorithm the same amount of time on each problem instance, allowing the algorithm to iterate as many times as possible within that time span[4]. In reality, as elaborated further below, allowing an algorithm longer and longer execution times beyond a modest base value appears to extend only very marginally the range of sizes of stable matchings generated. All implementations were in Java version 6.0 and were run on a 2.6 GHz PC with 500 Mb of RAM.

5.1 Real data

In the SFAS data for 2006 the numbers of residents, hospitals[5] and posts were 759, 53 and 801 respectively. In 2007, the corresponding figures were 781, 53 and 789, and in 2008 they were 748, 52 and 752. In all three cases, residents had (strictly ranked) preference lists of length 6 and the posts were fairly uniformly distributed across the hospitals. Hospitals had quite varied patterns of ties in their lists – ranging from strictly ranked lists in a few cases, to others where the residents were grouped in just two or three ties. As is to be expected in such circumstances, some hospitals, and some residents, were more popular than others. A plot of hospitals against number of applicants (i.e., the number of residents ranking each hospital) revealed that the the most popular had around five times the number of applicants as compared to the least popular, and the plot was close to linear between these two extremes. (This observation influenced the way that hospital popularity was modelled in the artificial data.)

We report the results of running each algorithm for the same length of time, namely 1 minute, on these three sets of data. This may seem like an unduly short time. In fact we also ran each algorithm on each set of data for 5 minutes, and for 25 minutes. Only in a very few cases did the observed distribution of stable matching sizes change and only in one case (Algorithm I – see below – on the 2006 data) did the size of the largest stable matching increase by as much as 2. In every case, the average size of stable matching found, rounded to one decimal place, did not change at all.

The results are summarised in Tables 1, 2 and 3. Here, H and R represent Algorithms HROST-Heuristic-H and HROST-Heuristic-R respectively, and K represents Király's algorithm. These abbreviations are used throughout this section. Algorithms I and C are both random tie-breaking algorithms; in Algorithm I, each tie is broken randomly and *independently*, whereas in Algorithm C, the residents are strictly ordered in a random way and every tie is broken *consistently* according to this ordering. For each algorithm and each data set, the table shows the number of residents involved, together with the maximum, minimum, mean and mode of the stable matching sizes found, and the number

[4]Further justification is provided by noting that any organisation running a matching scheme would seek to ensure that the matching program completed its execution within a reasonable, and preferably predictable, time frame.

[5]In SFAS, the terms 'students' and 'programmes' are used, rather than 'residents' and 'hospitals'.

of repetitions[6] in the allowed time. The spread of sizes of stable matchings found is also reported. In all tables, the highest value(s) in each column is highlighted in bold.

	2006 (759 residents)				
Algorithm	Max	Min	Mean	Mode	Iterations
H	753	**745**	748.8	**749**	4512
R	**754**	744	**749.3**	**749**	594
K	753	743	748.5	**749**	13366
I	744	726	736.5	736	14710
C	744	727	735.1	735	13961
Spread	726 - 754				

Table 1: Results on the SFAS data for 2006

	2007 (781 residents)				
Algorithm	Max	Min	Mean	Mode	Iterations
H	740	731	735.6	736	5838
R	**744**	**737**	**741.4**	**742**	766
K	742	732	736.9	737	13008
I	736	723	729.3	729	14538
C	736	722	728.9	729	14132
Spread	722 - 744				

Table 2: Results on the SFAS data for 2007

	2008 (748 residents)				
Algorithm	Max	Min	Mean	Mode	Iterations
H	**705**	**695**	699.5	699	33200
R	**705**	694	**700.2**	**700**	4120
K	**705**	**695**	700.1	**700**	61475
I	697	681	689.3	689	75470
C	697	680	688.5	689	75888
Spread	680 - 705				

Table 3: Results on the SFAS data for 2008

Commentary on the results

Perhaps not surprisingly, Algorithms H, R and K fare better than the random tie-breaking algorithms, with Algorithm R giving the best results overall, albeit by a small margin over the other two. The performance of Algorithm K is impressive, given its relative simplicity, and of course that algorithm has the additional advantage of a performance guarantee. There is very little to choose between the two random strategies, though independent tie-breaking seems to lead to slightly larger matchings. Also, the naïve tie-breaking algorithms give a significantly greater range of different matching sizes than Algorithms H, R and K. The differences may not seem dramatic. However, even if a sophisticated algorithm succeeds in matching only a handful of additional residents compared to the number matched by random tie-breaking, it could be a highly significant improvement for those

[6]Note that Algorithm R completes substantially fewer iterations than any of the other algorithms, presumably reflecting a worse average-case complexity.

individuals concerned, bearing in mind the impact that their assignment could have on their future careers. Moreover, for practical purposes, one could argue that an important comparison is between the mode size of stable matching found, say, by Algorithm I, and the maximum size found by, say, Algorithm R. This could be viewed as a measure of the potential benefit of, on the one hand allowing or encouraging ties in the hospitals' preference lists and employing a good heuristic with a moderate number of iterations, and on the other hand prohibiting ties (effectively forcing each hospital to break its own ties randomly) and using the classical Gale-Shapley algorithm. This difference averages 16.3 over the three sets of data.

Table 4 gives a more detailed breakdown of the numbers of stable matchings of various sizes found by the various algorithms for the 2006 data. The patterns for the 2007 and 2008 data sets were very similar.

Finally, we note that the only upper bounds that we have, with which to compare our results, are obtained merely by dropping the stability requirement and finding the size of a maximum cardinality matching of residents to hospitals. This size turns out to be 759, 781 and 745 for 2006, 2007 and 2008 respectively. In other words, in the first two years, all residents could have been matched in this way, and in 2008, all but three. However, evidence presented below (in the subsection "How close to optimal?") leads us to believe that the largest stable matchings found are a much closer approximation to optimal than suggested by these bounds.

2006	743	744	745	746	747	748	749	750	751	752	753	754
H	-	-	15	105	454	1119	1505	996	273	42	3	-
R	-	-	5	18	61	104	152	130	78	37	8	1
K	1	12	110	602	2040	3897	4048	2131	484	39	2	-
2006	727	728	729	730	731	732	733	734	735	736	737	738
I	1	2	11	41	122	322	755	1411	2147	2481	2680	2185
C	4	21	64	200	469	974	1583	2216	2478	2336	1731	1076
	739	740	741	742	743	744						
I	1421	721	294	93	21	2						
C	529	202	60	14	3	1						

Table 4: Matching size distributions for the SFAS data for 2006

5.2 Artificial data

In generating artificial data, a whole range of parameters and data characteristics can be varied, including

- the numbers of residents, hospitals, and posts;
- the lengths of residents' preference lists;
- the way in which posts are distributed among hospitals;
- the variation in hospital and resident popularity;
- the number, length and distribution of ties;
- whether, as in some real contexts, there is a 'master' preference list of residents (i.e., a global ranking of all residents) based on scores such as examination results; each hospital's preference list, as always, contains only the residents on whose list it appears, and the rank ordering of that subset of the residents is inherited from the rank ordering in the master list.

The number of possible combinations of these parameters and factors is huge, so we made a selection of test cases with a view to ascertaining how the variation of certain key factors would affect the performance of the algorithms and the variability of stable matching sizes. We fixed some parameters for reporting purposes – in all cases 1000 residents, 100 hospitals, 1000 posts and residents' preference lists of length 5. Further experiments showed that allowing these parameters to vary did not yield substantially different results, except in one respect – increasing (respectively decreasing) the ratio of posts to residents, or the lengths of preference lists, caused the sizes of stable matchings to increase (respectively decrease) across the board, with no significant differential behaviour among the various algorithms.

We allowed the posts either to be randomly or uniformly distributed among the hospitals. We allowed both residents and hospitals either to be uniformly popular or to exhibit a skewed popularity. Skewed hospital popularity was intended to simulate the variation observed in hospital popularity in SFAS (see above), with the most popular hospitals attracting about five times the number of applicants as the least popular, and the distribution approximately linear between these two extremes. Skewed resident popularity was envisaged to reflect the reality that some applicants are more sought after than others, and was implemented by introducing bias into the random number generation process that produced rank orderings of the hospitals' preference lists.

As far as ties were concerned, in the case of individual hospital preference lists, we varied the tie pattern by allowing the probability that an entry be tied with its successor to be either medium (0.5) or high (0.9). Note that, as might be expected, the closer the preference lists are to being strictly ordered, the less variation there is in the size of stable matchings, so that with low tie probability, the distinctions between the results of the different algorithms become less significant, and are not reported here. In the case of a master list of residents, a set of distinct 'scores' was fixed, and a random score allocated from this set to each resident. The master list was constructed on the basis of these scores, with equal scoring residents tied. Sets of 5 scores and 50 scores were used to correspond to high and medium tie probability, respectively.

In fact, it turned out, perhaps a little surprisingly, that skewed hospital popularity led to quite significant differences in outcomes, whereas skewed resident popularity (relevant only in the case of individual hospital preferences) merely reduced slightly the sizes of stable matchings observed, and appeared to affect all algorithms in more or less the same way. Of course, this may simply reflect the way that we chose to skew resident popularity during the data generation process. However, to reduce the volume of results reported here, only results for uniform resident popularity are presented. As in the case of the real data, each algorithm was given the same amount of time, again one minute, on each problem instance generated.

Individual hospital preference lists

Tables 5 and 6 show matching sizes for the five algorithms on sets of artificial data for the case of individual hospital preference lists. In each case the sizes represent an average taken over ten instances[7] generated with the parameter values in question. For each set of parameters, the *spread* shows the maximum, minimum and average difference between the largest and smallest stable matchings found over the ten instances. Note that the spread values shown in the various tables may well be underestimates of the true values, since it is likely that our algorithms often fail to find the largest and, more especially, the smallest

[7]This may seem like a small number of cases. However, even this small number of cases led to quite consistent results that would not have differed substantially had a larger number of cases been used.

stable matchings. Indeed, algorithms analogous to Algorithms H, K and R but designed to find smaller, rather than larger, stable matchings, could have been implemented, and had this been done, significantly smaller stable matchings than those returned by our random tie-breaking algorithms might well have been found.

Master list of residents

Tables 7 and 8 show matching sizes for the five algorithms on sets of artificial data for the case of a master list of residents, constructed on the basis of scores assigned to the residents. Table 7 summarises the case where scores were assigned randomly from 50 possibilities, and Table 8 for the case of just 5 possibilities (so that ties were typically much longer in the latter case). Again, in each case the size represents an average taken over ten instances generated with each set of parameter values, and the spread values indicate the maximum, minimum and average range of stable matching sizes found, taken over the ten instances. As before, each algorithm was run for one minute on each generated instance.

	Random post distrib. Uniform hos. pop.			Random post distrib. Skewed hos. pop.			Uniform post distrib. Uniform hos. pop.			Uniform post distrib. Skewed hos. pop.		
Alg	Max	Min	Ave	Max	Min	Ave	Max	Min	Ave	Max	Min	Ave
H	**990.0**	983.9	986.9	961.5	954.6	958.2	996.3	991.3	993.9	975.0	968.7	971.7
R	989.8	**986.1**	**988.1**	**961.8**	**958.1**	**960.1**	**996.6**	**993.0**	**994.9**	**975.2**	**971.8**	**973.5**
K	989.9	983.6	986.9	961.5	954.3	958.1	996.4	991.1	994.0	975.0	968.5	971.7
I	989.2	981.5	985.4	960.2	951.2	955.7	996.0	989.5	992.8	974.3	965.7	969.9
C	989.1	981.6	985.3	960.5	951.0	955.7	995.8	989.5	992.8	974.2	965.6	969.9
Spread	11	7	8.4	15	8	11.0	8	6	7.1	13	5	9.6

Table 5: Matching size distributions for HROST artificial data I: medium tie probability

	Random post distrib. Uniform hos. pop.			Random post distrib. Skewed hos. pop.			Uniform post distrib. Uniform hos. pop.			Uniform post distrib. Skewed hos. pop.		
Alg	Max	Min	Ave	Max	Min	Ave	Max	Min	Ave	Max	Min	Ave
H	998.2	990.9	994.7	986.5	976.2	981.5	999.6	995.3	997.9	990.0	979.8	985.0
R	**998.3**	**994.7**	**996.7**	**988.3**	**980.7**	**984.5**	**999.9**	**998.2**	**999.3**	**993.2**	**986.3**	**990.2**
K	998.0	990.3	994.6	986.2	974.2	980.5	**999.9**	994.9	997.9	990.1	978.9	984.7
I	992.3	977.4	985.3	972.1	952.5	962.5	998.2	986.8	993.0	979.4	961.2	970.8
C	991.3	974.3	983.2	969.8	949.4	959.7	997.3	984.9	991.4	977.5	959.0	968.6
Spread	30	20	24.0	50	32	39.0	18	12	15.0	44	30	34.2

Table 6: Matching size distributions for HROST artificial data II: high tie probability

	Random post distrib. Uniform hos. pop.			Random post distrib. Skewed hos. pop.			Uniform post distrib. Uniform hos. pop.			Uniform post distrib. Skewed hos. pop.		
Alg	Max	Min	Ave	Max	Min	Ave	Max	Min	Ave	Max	Min	Ave
H	**966.7**	962.7	964.7	931.1	927.6	929.2	**977.5**	973.5	975.6	**944.4**	940.3	942.3
R	966.3	**965.8**	**966.0**	930.4	**929.3**	**929.9**	976.9	**976.6**	**976.7**	944.0	**943.4**	**943.7**
K	**966.7**	962.7	964.7	**931.2**	927.4	929.1	**977.5**	973.1	975.4	**944.4**	940.1	942.3
I	966.6	959.5	963.1	931.0	924.1	927.6	977.4	970.7	974.0	944.3	938.0	941.0
C	**966.7**	959.6	963.1	930.8	924.1	927.6	977.4	970.7	974.0	944.3	938.0	941.0
Spread	9	5	7.1	11	3	6.3	10	3	6.8	8	2	6.4

Table 7: Matching size distributions for HROST artificial data III: master list of residents, high number of scores

	Random post distrib. Uniform hos. pop.			Random post distrib. Skewed hos. pop.			Uniform post distrib. Uniform hos. pop.			Uniform post distrib. Skewed hos. pop.		
Alg	Max	Min	Ave	Max	Min	Ave	Max	Min	Ave	Max	Min	Ave
H	987.0	975.3	981.5	958.2	944.0	951.3	997.1	988.8	993.3	966.8	953.0	959.9
R	**993.5**	**993.0**	**993.1**	**966.5**	**961.3**	**963.9**	**998.9**	**998.9**	**998.9**	**975.8**	**973.2**	**974.4**
K	987.5	974.6	981.3	958.2	943.6	951.0	998.0	988.3	993.3	967.5	952.9	960.2
I	973.0	953.5	963.6	945.3	923.8	935.1	985.5	966.2	976.2	955.0	934.7	945.0
C	972.1	951.5	961.8	934.6	923.4	933.8	983.1	962.7	973.3	953.3	933.2	943.6
Spread	52	35	42.0	52	38	43.1	41	31	36.2	50	36	42.6

Table 8: Matching size distributions for HROST artificial data IV: master list of residents, low number of scores

Empirical results for the Stable Marriage problem

Although of less practical significance, it is still of interest to investigate the behaviour of the various algorithms on instances of the Stable Marriage problem. We studied the special case, SMOSTI, corresponding to instances of HROST in which each hospital has capacity 1, and in which men (respectively women) play the role of residents (respectively hospitals).

Tables 9 and 10 show empirical results for this special case based on the same five algorithms. As in the HROST case, varying the popularity pattern of the agents appearing in ties (in this case the men) had a barely perceptible effect on the results, so we do not include this information in the tables. In all cases, we assumed 1000 men, 1000 women, and men's preference lists of length 5. As before, each algorithm was run for one minute on each generated instance.

How close to optimal?

One obvious key question arises from this empirical study, namely how close to optimal are the stable matchings produced by our various algorithms? Of course, not knowing the size of a maximum stable matching makes it hard to answer this question, and as mentioned earlier, the only upper bound is that obtained from a maximum cardinality matching, which in many cases we suspect to be a weak bound.

We carried out some additional experiments to investigate this issue. We generated artificial instances of both HROST and SMOSTI in which a complete stable matching (i.e. a stable matching including all of the participants) was 'planted'. As in our earlier experiments, the HROST instances had 1000 residents, 100 hospitals, and 1000 posts, with the posts either uniformly or randomly distributed among the hospitals, and residents' preference lists of length 5. The SMOSTI instances had 1000 men and 1000 women with men's preference lists of length 5. We were again able to vary hospital popularity and tie structure, though we handled the latter aspect in a slightly different way as compared

	Medium tie prob. Uniform woman pop.			Medium tie prob. Skewed woman pop.			High tie prob. Uniform woman pop.			High tie prob. Skewed woman pop.		
Alg	Max	Min	Ave	Max	Min	Ave	Max	Min	Ave	Max	Min	Ave
H	947.6	933.1	940.2	924.0	910.7	917.4	**991.4**	**990.7**	**991.0**	**973.8**	**973.2**	**973.5**
R	**959.9**	**955.9**	**957.6**	**931.9**	**936.5**	**928.9**	989.3	987.6	988.0	971.6	967.5	969.3
K	948.4	930.9	940.0	924.6	906.5	915.6	981.4	964.8	973.6	964.3	944.6	954.9
I	924.7	897.2	910.7	898.7	870.9	884.9	925.0	892.6	909.0	904.3	869.3	887.0
C	922.1	894.0	908.0	895.5	869.5	882.5	908.9	874.4	892.5	887.6	851.0	870.5
Spread	72	59	66.0	67	55	62.4	127	113	117.0	129	113	122.8

Table 9: Matching size distributions for SMOSTI artificial data I: individual preference lists

	High number of scores Uniform woman pop.			High number of scores Skewed woman pop.			Low number of scores Uniform woman pop.			Low number of scores Skewed woman pop.		
Alg	Max	Min	Ave	Max	Min	Ave	Max	Min	Ave	Max	Min	Ave
H	**884.6**	880.3	882.4	863.2	858.6	860.9	905.7	888.8	896.9	881.5	865.1	873.1
R	**884.6**	**884.5**	**884.5**	**863.3**	**863.3**	**863.3**	**919.4**	**917.8**	**918.4**	**894.4**	**891.1**	**892.5**
K	**884.6**	880.1	882.4	863.2	858.6	860.9	907.8	889.0	898.5	883.2	864.9	874.3
I	**884.6**	877.6	881.1	863.1	855.7	859.4	895.0	870.7	883.2	871.1	848.3	860.1
C	884.4	877.7	881.1	863.1	855.7	859.3	894.1	870.5	882.4	870.9	847.8	859.4
Spread	10	3	7.0	14	4	7.7	53	44	49.1	54	42	46.8

Table 10: Matching size distributions for SMOSTI artificial data II: master preference list

to our earlier experiments. In order to ensure the stability of the planted matching, it was easier to construct each hospital's preference lists as a sequence of ties, each tie corresponding to a numerical 'score'. The *score range* therefore gives an upper bound on the number of ties per list. Entries were distributed among the ties randomly, subject to preservation of the stability of the planted matching. An additional parameter that can be varied here is the *expected rank* in a resident r_i's preference list of her assigned hospital h_j in the planted matching (the *rank* of h_j in r_i's list is the k such that h_j is r_i's kth choice). The value of this parameter determines how residents' preferences are constructed – the planted matching is inserted first, then the other entries are placed before or after it with appropriate probabilities.

As before, we found that uniform popularity, uniform post distribution and fewer tied entries led to less distinction among the algorithms, so we focused on cases of skewed popularity, random post distribution and relatively small score ranges.

Tables 11 and 12 show the maximum, minimum and average size of stable matchings found by the five algorithms for the various combinations of score range and expected matching rank. These are averages of ten generated HROST instances in each case, and every run was of one minute's duration. Tables 13 and 14 give the corresponding results for instances of SMOSTI.

Discussion

A number of general conclusions can be drawn from these experiments, including the following:

- Algorithms H, R and K consistently find larger stable matchings than Algorithms I and C. This is more noticeable when the tie probability is high, and is accentuated when hospital popularity is skewed and when posts are distributed randomly rather than uniformly. The rationale for this observed behaviour is that the first three algorithms all use tie-breaking strategies that, in some way, give priority to residents who would otherwise have a higher likelihood of being left unmatched.

- Algorithm R typically finds slightly larger stable matchings than Algorithms H and K, especially when tie probability is high. For every single set of artificial test data, Algorithm R performed best in terms of the average and smallest size of stable matchings. The superiority of Algorithm R is most marked in the case of master lists with a low number of scores; in these cases the smallest matching found by Algorithm R is larger than the largest matching found by Algorithms H or K, and the range of matching sizes found by Algorithm R is often very small. Algorithm R appears to be marginally less successful at finding a largest matching when there is a master list based on a high number of scores, but in any case the spread of matching sizes in these circumstances is quite small.

	score range = 10						score range = 5					
	expected rank = 2			expected rank = 3			expected rank = 2			expected rank = 3		
Alg	Max	Min	Ave	Max	Min	Ave	Max	Min	Ave	Max	Min	Ave
H	**1000**	997.0	999.5	**1000**	998.9	**1000**	**1000**	997.8	999.8	**1000**	998.4	**1000**
R	**1000**	**999.4**	**999.9**	**1000**	**1000**	**1000**	**1000**	**999.9**	**1000**	**1000**	**1000**	**1000**
K	**1000**	996.9	999.5	**1000**	998.5	**1000**	**1000**	997.6	999.8	**1000**	998.2	**1000**
I	**1000**	994.7	998.1	999.9	991.6	996.7	999.9	992.6	997.3	999.4	987.4	994.1
C	**1000**	993.9	997.8	999.7	989.6	995.2	999.7	991.2	996.3	997.6	983.2	991.1

Table 11: HROST instances with a planted stable matching of size 1000

	score range = 3						score range = 2					
	expected rank = 2			expected rank = 3			expected rank = 2			expected rank = 3		
Alg	Max	Min	Ave	Max	Min	Ave	Max	Min	Ave	Max	Min	Ave
H	**1000**	998.2	999.9	**1000**	998.9	**1000**	**1000**	998.8	**1000**	**1000**	999.0	**1000**
R	**1000**	**1000**	**1000**	**1000**	**1000**	**1000**	**1000**	**1000**	**1000**	**1000**	**1000**	**1000**
K	**1000**	997.8	999.9	**1000**	998.2	**1000**	**1000**	998.0	999.9	**1000**	998.2	**1000**
I	999.8	989.5	995.5	997.7	981.9	990.5	998.7	985.0	992.8	995.0	976.1	986.6
C	998.5	985.2	992.8	993.1	974.1	984.3	994.8	977.0	986.5	987.8	962.7	975.8

Table 12: HROST instances with a planted stable matching of size 1000

	score range = 10						score range = 5					
	expected rank = 2			expected rank = 3			expected rank = 2			expected rank = 3		
Alg	Max	Min	Ave	Max	Min	Ave	Max	Min	Ave	Max	Min	Ave
H	981.2	965.6	973.3	975.7	958.8	967.3	979.4	963.2	971.3	974.5	958.2	966.6
R	**994.0**	**992.0**	**992.9**	**989.9**	**984.9**	**987.4**	**995.1**	**993.6**	**994.3**	**993.3**	**988.5**	**990.6**
K	982.0	965.8	974.1	977.0	958.8	968.2	982.2	964.3	973.5	977.6	957.8	967.9
I	972.7	950.2	961.4	956.2	930.1	943.4	963.1	938.6	951.0	949.2	919.4	934.3
C	972.0	948.6	960.5	956.2	930.1	943.4	960.8	935.3	948.4	944.7	915.2	929.9

Table 13: SMOSTI instances with a planted stable matching of size 1000

	score range = 3						score range = 2					
	expected rank = 2			expected rank = 3			expected rank = 2			expected rank = 3		
Alg	Max	Min	Ave	Max	Min	Ave	Max	Min	Ave	Max	Min	Ave
H	982.3	967.3	974.7	980.7	965.7	973.3	990.3	978.9	984.8	990.9	981.0	985.7
R	**997.8**	**996.1**	**996.7**	**994.8**	**991.4**	**992.8**	**999.1**	**998.8**	**998.8**	**997.2**	**994.8**	**995.7**
K	984.3	966.3	975.8	980.8	961.8	971.5	987.6	970.8	980.0	983.5	965.3	974.9
I	957.9	930.2	943.9	941.9	910.3	926.2	950.9	920.8	936.5	935.7	903.0	919.6
C	952.1	924.0	938.8	932.8	903.0	918.4	942.6	910.6	926.5	923.3	890.5	907.4

Table 14: SMOSTI instances with a planted stable matching of size 1000

- Algorithm K performs remarkably well, and has the advantage of conceptual simplicity as well as a performance guarantee. All the same, for none of the sets of test data that we used did this guarantee provide more information than could have been deduced from knowing that the size of a maximum stable matching cannot exceed the number of applicants.

- It is hard to come up with an explanation as to why Algorithm R appears to be marginally superior to the others. However, in comparison with the cardinality matching approach of Algorithm H, it could be argued that the construction of the network and the exploitation of network flow allows Algorithm R to look further ahead in considering the consequences of a particular tie-break.

- Algorithm C is usually marginally less effective than Algorithm I. But the difference is quite noticeable in the case of SMOSTI with individual preference lists and high tie probability. Some intuition for this comes from observing that uniform tie-breaking leads to strictly ordered preference lists that are somewhat more correlated than those resulting from independent tie-breaking, and greater similarity among preferences can be expected to reduce the size of a stable matching.

- Not surprisingly, the spread of matching sizes appears to be greater when there are longer ties, and can be very substantial in such cases. The more tied entries there are, the higher will be the proportion of all possible matchings that are stable.

- The outcomes for SMOSTI shown in Tables 9 and 10 are broadly similar to those for HROST shown in Tables 5-8. An exception is in the case of individual preference lists when tie probability is high, Algorithm H being clearly the most effective strategy in this case. This can be explained by observing that many women's preference lists in this case will consist of a single tie, and the cardinality matching strategy of Algorithm H will be very effective in this situation. The spread of matching sizes appears to be typically greater for SMOSTI.

- In the case where we planted a complete stable matching, and therefore knew the size of the optimal solution, for all of the instances of HROST that we generated, at least one of Algorithms H, R or K (in most cases all three of them) found a stable matching of maximum size. On the other hand, Algorithms I and C hardly ever found such a stable matching, often falling well short, depending on the parameter values. Of course, it may be that planting a complete stable matching somehow distorts the problem instance, and it may not be entirely valid to extrapolate this observed behaviour to other instances of HROST. Nonetheless, we are left with some increased confidence that, at least in the case of our SFAS examples, the largest stable matching that we found may be quite close to optimal.

- However, in this respect, instances of SMOSTI revealed a strikingly different story. The range of sizes of stable matchings found was much greater, and in most cases none of the algorithms found a complete stable matching. For example, in one of the more extreme instances, the ranges of stable matching sizes produced by algorithms H, R, K, I and C were 957-974, 990-991, 961-978, 933-956 and 922-954 respectively. It is quite noticeable here that, in every single instance that we generated, Algorithm R found a larger stable matching than any of the other algorithms, often by a considerable margin.

6 Summary and open problems

This paper has presented an empirical investigation of heuristics for finding a maximum stable matching in instances of the Stable Marriage and Hospitals/Residents problems with ties on one side. Two new heuristics have been proposed and their performance compared with Király's approximation algorithm, the algorithm for these problems with the best known performance guarantee, and with two random tie-breaking strategies. The performance of these various algorithms has been studied on some limited real data, and on artificial data generated according to a number of models. One of these new heuristics, Algorithm HROST-Heuristic-R, which combines the classical Gale-Shapley algorithm with a step based on network flow, was found in most cases to be the most effective method. Evidence of the considerable spread of possible stable matching sizes was a by-product of these experiments.

As mentioned in Section 2, the hospital-oriented heuristic is an extension of the approximation algorithm appearing in [14]. The latter has a performance guarantee of 5/3 when ties are restricted to one per list, and can occur only at the end of the list, so the guarantee for that special case applies also to the hospital-oriented heuristic. This paper is primarily concerned with the performance of the algorithms for MAX-HROST and MAX-SMOSTI in practice, however it would be worth carrying out a closer analysis of Algorithms HROST-Heuristic-H and HROST-Heuristic-R, to determine whether a concrete worst-case performance guarantee better than 2 can be established for either. In addition, it would be interesting to try to construct instances for which the heuristics perform particularly badly, perhaps approaching the worst-case approximation ratio of 2.

It could be argued that it is inappropriate to focus exclusively on the size of a stable matching while ignoring other possibly desirable properties. It is worth observing that Erdil and Ergin [5] have recently introduced the concept of a *stable improvement cycle* in an attempt to improve the quality of a stable matching from the residents' viewpoint. Locating and applying stable improvement cycles allows an arbitrary stable matching to be transformed to another of the same size which is Pareto optimal for the residents. Hence this process, which is straightforward to implement efficiently, could form a final step in any heuristic employed in a practical setting.

Acknowledgment

We are grateful to the reviewers for some helpful suggestions that led to significant improvements in the presentation of the paper.

References

[1] A. Abdulkadiroğlu, P.A. Pathak, and A.E. Roth. The New York City high school match. *American Economic Review*, 95(2):364–367, 2006.

[2] A. Abdulkadiroğlu, P.A. Pathak, A.E. Roth, and T. Sönmez. Changing the Boston school-choice mechanism. NBER working paper 11965, 2006.

[3] M. Baïou and M. Balinski. Student admissions and faculty recruitment. *Theoretical Computer Science*, 322(2):245–265, 2004.

[4] P. Biró. Higher education admission in Hungary by a score-limit algorithm. In *Proceedings of the 18th International Conference on Game Theory, Stony Brook.* `http://www.cs.bme.hu/~pbiro/college_admission.pdf`, 2007.

[5] A. Erdil and H. Erkin. What's the matter with tie-breaking? Improving efficiency in school choice. *American Economic Review*, 98:669–689, 2008.

[6] H.N. Gabow. An efficient reduction technique for degree-constrained subgraph and bidirected network flow problems. In *Proceedings of STOC '83: the 15th Annual ACM Symposium on Theory of Computing*, pages 448–456. ACM, 1983.

[7] D. Gale and L.S. Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 69:9–15, 1962.

[8] D. Gale and M. Sotomayor. Some remarks on the stable matching problem. *Discrete Applied Mathematics*, 11:223–232, 1985.

[9] D. Gusfield and R.W. Irving. *The Stable Marriage Problem: Structure and Algorithms*. MIT Press, 1989.

[10] M. Halldórsson, K. Iwama, S. Miyazaki, and H. Yanagisawa. Improved approximation of the stable marriage problem. *ACM Transactions on Algorithms*, 3(3), 2007. Article number 30.

[11] M.M. Halldórsson, K. Iwama, S. Miyazaki, and H. Yanagisawa. Randomized approximation of the stable marriage problem. *Theoretical Computer Science*, 325(3):439–465, 2004.

[12] R.W. Irving. Stable marriage and indifference. *Discrete Applied Mathematics*, 48:261–272, 1994.

[13] R.W. Irving and P. Leather. The complexity of counting stable marriages. *SIAM Journal on Computing*, 15(3):655–667, 1986.

[14] R.W. Irving and D.F. Manlove. Approximation algorithms for hard variants of the stable marriage and hospitals/residents problems. *Journal of Combinatorial Optimization*, 16(3):279–292, 2008.

[15] R.W. Irving, D.F. Manlove, and G. O'Malley. Stable marriage with ties and bounded length preference lists. *Journal of Discrete Algorithms,* to appear, 2009.

[16] K. Iwama, S. Miyazaki, and K. Okamoto. A $\left(2 - c\frac{\log n}{n}\right)$-approximation algorithm for the stable marriage problem. In *Proceedings of SWAT 2004: the 9th Scandinavian Workshop on Algorithm Theory*, volume 3111 of *Lecture Notes in Computer Science*, pages 349–361. Springer, 2004.

[17] K. Iwama, S. Miyazaki, and N. Yamauchi. A $\left(2 - c\frac{1}{\sqrt{n}}\right)$-approximation algorithm for the stable marriage problem. In *Proceedings of ISAAC 2005: the 16th Annual International Symposium on Algorithms and Computation*, volume 3827 of *Lecture Notes in Computer Science*, pages 902–914. Springer, 2005.

[18] K. Iwama, S. Miyazaki, and N. Yamauchi. A 1.875–approximation algorithm for the stable marriage problem. In *Proceedings of SODA 2007: the Eighteenth ACM/SIAM Symposium on Discrete Algorithms*, pages 288–297, 2007.

[19] Z. Király. Better and simpler approximation algorithms for the stable marriage problem. In *Proceedings of ESA '08: the 16th Annual European Symposium on Algorithms,* to appear, Lecture Notes in Computer Science, 2008.

[20] D.F. Manlove, R.W. Irving, K. Iwama, S. Miyazaki, and Y. Morita. Hard variants of stable marriage. *Theoretical Computer Science*, 276(1-2):261–279, 2002.

[21] E. McDermid. A 3/2 approximation algorithm for general stable marriage. Technical Report TR-2008-287, University of Glasgow, Department of Computing Science, 2008.

[22] A. Romero-Medina. Implementation of stable solutions in a restricted matching market. *Review of Economic Design*, 3(2):137–147, 1998.

[23] A.E. Roth. The evolution of the labor market for medical interns and residents: a case study in game theory. *Journal of Political Economy*, 92(6):991–1016, 1984.

[24] `http://www.bostonpublicschools.org/assignment/` (Boston Public Schools website.

[25] `http://www.carms.ca` (Canadian Resident Matching Service website).

[26] `http://www.nrmp.org` (National Resident Matching Program website).

[27] `http://www.nes.scot.nhs.uk/sfas` (Scottish Foundation Allocation Scheme website).