MASARYK UNIVERSITY
FACULTY OF INFORMATICS

Stable Marriage Problem Algorithms

MASTER THESIS

Andrej Podhradský

Brno, Spring 2010

Declaration

Hereby I declare, that this paper is my original authorial work, which I have worked out by my own. All sources, references and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Advisor: prof. RNDr. Ivana Černá, CSc.

Abstract

The goal of this thesis is to clearly present approximation algorithms for the MAX SMTI problem. Selected algorithms are implemented and practicly compared.

Keywords

stable marriage, max smti, approximation algorithm, indifference, ties, incomplete list

Contents

Chapter 1

Introduction

1.1 Stable Marriage Problem

The Stable Marriage Problem is a matching problem in graph theory first introduced by Gale and Shapley [6]. An instance of this problem consists of two sets, a set of men and a set of women. Each person specifies the order of all members of opposite sex. A matching is stable if there is no pair such that both man and woman prefers each other to their current partner in the matching. The goal of this problem is to find a stable matching for a given instance. Gale and Shapley show that there always exists such a matching for any instance. They also provide a polynomial algorithm for solving the problem.

Practical applications use a generalized variant of the Stable Marriage Problem called the Hospital/Residents Problem. In this variant more residents can be assigned to one hospital. The Hospital/Residents Problem is used in matching schemes, such as National Resident Matching Program [3], the Canadian Resident Matching Service [1] and the Scotish Pre-registration house officer Allocations schema [4].

1.2 Our Contribution

The classical Stable Marriage Problem has a polynomial solution. But if we allow incomplete preference lists and ties a new problem arises, called MAX SMTI. The goal of this problem is to find a maximum stable matching. In fact, this problem is NP-hard [18].

In this thesis we describe some approximation algorithms for MAX SMTI problem. Then, we implement the algorithms which are available from the web via an applet. Subsequently, we compare these algorithms on a large scale inputs.

1.3 Thesis Organization

In Chapter 2 we define the Stable Marriage Problem, describe G-S algorithm and its properties. Then, we present some generalization of the problem, especially a variation with incomplete preference lists and ties (SMTI).

In Chapter 3 we introduce MAX SMTI problem which is NP-hard problem. We provide some inapproximability results and discuss how to obtain an optimal solution.

In Chapter 4 we clearly describe and explain some approximation algorithms for MAX SMTI problem. Then, we implement these algorithms and put some implementation remarks into Chapter 5.

Chapter 6 summarizes our algorithm comparisons, results and observations.

Chapter 2

Stable Marriage Problem

This chapter describes the Stable Marriage Problem and an algorithm that solves it. Then, we introduce some variants of this problem.

2.1 Definition

Definition 2.1. An instance of the Stable Marriage Problem (SM) consists of a set of n men and a set of n women where each man and each woman provides a lineary ordered lists of all members of the opposite sex. Such lists are called *preference* lists.

Let M be an one-to-one correspondence between the subset of men and the subset of women, then we say that M is a *marriage*. Let $(m, w) \in M$, then we say that (m, w) is a *pair* in M or that m is married with w in M or that w is *married* with m in M. Furthermore, we say that a man m (a woman w) is free in M if he (she) is not married with any woman (man) in M, otherwise we say that m (w) is matched in M. Let $M(m)$ denote a woman who is *married* with a man m in M and let $M(w)$ denote a man married with w in M.

Let p, q, q' be persons such that q, q' are of the same sex and p is of opposite sex. We say that *p prefers q* to q' if q appears at the left side from q' in p's preference list, denoted as $q \succ_p q'$.

Let X be a set of men, let Y be a set of women and let $M \subset X \times Y$ be a marriage. A pair $(m, w) \in X \times Y$ is a *blocking* pair for M if $(m, w) \notin$ and at least one of the following conditions is satisfied:

- both m and w are free in M
- m is free in M and w prefers m to $M(w)$
- w is free in M and m prefers w to $M(m)$

If there is no blocking pair for M then we say that M a *stable marriage*.

The goal of the Stable Marriage Problem is to find a stable marriage for an instance of SM. An example of the instance of SM is depicted on Figure 6.15. In this example the first part refers to men's preference lists and the second one refers to women' preference lists. Each line is parsed into an owner (before the colon) and a preference list itself (after

Men's list								
1:	5	7	1	2	6	8	4	3
2:	2	3	7	5	4	1	8	6
3:	8	5	1	4	6	2	3	7
4:	3	2	7	4	1	6	8	5
5:	7	2	5	1	3	6	8	4
6:	1	6	7	5	8	4	2	3
7:	2	5	7	6	3	4	8	1
8:	3	8	4	5	7	2	6	1

Women's list								
1:	5	3	7	6	1	2	8	4
2:	8	6	3	5	7	2	1	4
3:	1	5	6	2	4	8	7	3
4:	8	7	3	2	4	1	5	6
5:	6	4	7	3	8	1	2	5
6:	2	8	5	4	6	3	7	1
7:	7	5	2	1	8	6	4	3
8:	7	4	1	5	2	3	6	8

Figure 2.1: Male and female preference lists

the colon). The most preffered person is placed to the leftmost of the preference list. An example of stable marriage M for such instance is as follows:

$$M = \{(1,5),(2,3),(3,8),(4,6),(5,7),(6,1),(7,2),(8,4)\}$$

Gale and Shepley [6] proved that for every instance of SM there exists at least one stable marriage. They also gave an Algorithm 2.1 which always finishes and runs in $O(n^2)$. We call this algorithm the *G-S algorithm*. The algorithm is based on iteratively making proposals by a free man.

Algorithm 2.1 G-S [6]

```
Input: an instance of SM
Output: a stable marriage $M$
 1: while there is a free man $m$ who hasn't proposed to every women do
 2:     $m$ proposes to the most preffered woman $w$ he hasn't yet proposed to
 3:     if $w$ is free then
 4:         $(m, w)$ becomes engaged
 5:     end if
 6:     if $w$ prefers $m$ to her current partner $M(w)$ then
 7:         break the pair $(M(w), m)$                    // $w$ and $M(w)$ are free
 8:         $(m, w)$ becomes engaged
 9:     end if
10: end while
```

Male and female optimal stable marriage

The G-S algorithm finds a stable marriage which is optimal for men (*male-optimal*) since firtsly a man proposes to woman he most prefers and the woman just accepts the best from available proposals. If we interchange men with women we get a *female-optimal* solution. This observation is expressed in Lemma 2.2. From this lemma it is obvious

$$\begin{aligned}
M_1 &= \{(1,5),(2,3),(3,8),(4,6),(5,7),(6,1),(7,2),(8,4)\}\\
M_2 &= \{(1,8),(2,3),(3,5),(4,6),(5,7),(6,1),(7,2),(8,4)\}\\
M_3 &= \{(1,3),(2,6),(3,5),(4,8),(5,7),(6,1),(7,2),(8,4)\}\\
M_4 &= \{(1,3),(2,6),(3,1),(4,8),(5,7),(6,5),(7,2),(8,4)\}\\
M_5 &= \{(1,3),(2,6),(3,2),(4,8),(5,1),(6,5),(7,7),(8,4)\}\\
M_6 &= \{(1,3),(2,6),(3,1),(4,8),(5,2),(6,5),(7,7),(8,4)\}\\
M_7 &= \{(1,8),(2,3),(3,1),(4,6),(5,7),(6,5),(7,2),(8,4)\}\\
M_8 &= \{(1,8),(2,3),(3,2),(4,6),(5,1),(6,5),(7,7),(8,4)\}\\
M_9 &= \{(1,8),(2,3),(3,1),(4,6),(5,2),(6,5),(7,7),(8,4)\}
\end{aligned}$$

Figure 2.2: All stable marriages for an instance of SM depicted on Figure 6.15

that if another stable mariage exists then at least one man has a worse mate even as at least one woman has a better mate.

Lemma 2.2 ([6]). *For every man and woman the following two conditions hold for every stable marriage:*

- *each man is not engaged to a better woman than in the male optimal stable marriage*
- *each woman is not engaged to a worse man than in the male optimal stable marriage*

All stable marriages for an instance of SM depicted on Figure 6.15 are shown at Figure 2.2 where M_1 is a male optimal stable marriage and M_5 is a female optimal stable marriage.

2.2 Generalization of the Stable Marriage Problem

The original Stable Marriage Problem is very restrictive for practical applications so it is natural to consider about variations that are not so restrictive.

Incomplete lists

The first generalization of SM is that some partners may find other partners unacceptable. In this case we enable that preferece lists may be incomplete.

Definition 2.3. Let p, q be persons of opposite sex, then we say that p is *acceptable* for q if p appears on the preference list of q, otherwise p is *unacceptable* for q. Let *SMI* denote such a generalizaion of SM where incomplete preference lists are allowed.

Definition 2.4. Given an instance of SMI. A marriage M is stable if there is no unmatched pair (p, q) such that:

1. p and q is unmatched in M and finds each other acceptable
2. or each prefers the other to his/her partner in M.

Indifference

We can also consider a situation when some partner does not distinguish between two or more partners. In this case we allow that partners do not have to rank all partners of opposite sex in strict order, the preference list may contain ties. Let SMT denote such a generalization of SM.

Definition 2.5. Given an instance of SMT. Let p, q, q' be persons such that q, q' are of the sme sex and p is of opposite sex. We say that *p strictly prefers q* to q' if p prefers q to q' and q is not tied with q'. On the other side we say that *p prefers q* to q' if p prefers q or q and q' are in a tie. A marriage M is [10]:

1. *weakly stable* if there is no pair (p, q), each of whom strictly prefers the other to his/her partner in M
2. *strongly stable* if there is no pair (p, q) such that one of them strictly prefers the other to his/her partner in M and the second one prefers the other to his/her partner in M
3. *super-stable* if there is no pair (p, q), each of whom strictly prefers the other to his/her partner in M.

Such pairs are also called *blocking pairs*.

Notation 2.6. The ties are shown in parenthesis, e.g.: $w_1 :\ m_2\ (m_1\ m_3)\ m_4$ and costs are as follows $c(w) = 1, 2, 2, 4$ for m_1, m_2, m_3, m_4, respectively.

Note that weakly stable marriage for an instance of SMT always exists and it can be found easily in $O(n^2)$ by breaking ties in arbitralily way. On the other side, there are instances of SMT for which strongly or super stable marriages can not be found. Irving [10] presents $O(n^4)$ and $O(n^2)$ algorithms for determining if strongly and super stable marriages exist, respectively. Providing strongly or super stable marriage exists, the algorithms find them..

Ties and incomplete lists

Now, we consider the last generalization which is combination of SMI and SMT. Such a variant with incomplete list and indifference we mark as SMPTI.

Definition 2.7. Given an instance of SMTI. A marriage M is:

1. *weakly stable* if there is no unmatched pair (p, q) and finds each other acceptable or each of them strictly prefers the other to his/her partner in M
2. *strongly stable* if there is no unmatched pair (p, q) such that
   (a) either p is unmatched in M and finds q accaptable, or p strictly prefers q to his/her partner in M

An instance of SMTI				Weakly stable matchings
m_1:	w_1	w_1 :	(m_1, m_2)	$\{(m_1, w_1), (m_2, w_2)\}$
m_2:	$w_1 w_2$	w_2 :	m_2	$\{(m_2, w_1)\}$

Figure 2.3: An instance of SMTI with two weakly stable matchings of different size.

(b) and either q is unmatched in M and finds p accaptable, or p strictly prefers q to his/her partner in M, or q is indifferent between p and his/her partner in M

3. *super-stable* if there is no unmatched pair (p, q), each of whom is either unmatched in M and finds the other accaptable, or strictly prefers the other to his/her partner in M, or is indifferent between them.

It is not hard to determine if for an instance of SMTI there exists a strongly stable or super-stable matching. Manlove [18] presents algorithms that solve such problems and run in time $O(n^4)$ and $O(n^2)$, respectively. The algorithms also find such a matching if there any exists.

All strongly and super-stable matchings for an instance of SMTI are of the same size [18], but all weakly stable matchings need not be of the same size as it is depicted on Figure 2.3.

Chapter 3

MAX SMTI

This chapter introduces new problem MAX SMTI that arises when we allow ties and incomlete lists in an instance of SM. At first, we define the problem and then we describe ways how to solve it and finally we present some inapproximability results since the problem is NP-hard.

3.1 Definition

[18] Since there might exist stable marriages of different size for an instance of SMTI (Figure 2.3) we define the following decision problem:

Name: WEAK STABILITY SMTI
Instance: An instance of SMTI and integer $K \in \mathbb{Z}^+$
Question: Is there a weakly stable marriage M with $|M| \geq K$ for a given instance?

Now, we indtroduce its appropriate optimization problem as follows:

Name: MAX SMTI
Instance: An instance of SMTI
Purpose: Find a weakly stable marriage of maximum cardinality for a given instance.

Notation 3.1. In the following text we use stable marriage instead of weakly stable marriage.

3.2 NP-hardness and inapproximability results

In this section we describe all known results about inapproximability of the MAX SMTI.

It is proved that MAX SMTI is NP-hard [18, 13] and Manlove et al. [17] also show that it is NP-hard even for very restricted cases when ties o. Moreover, MAX SMTI is APX-hard, e.g. it is hard to approximate within $1 + \epsilon$, for some $\epsilon > 0$, even if ties appear in only men's lists, each man writes at most one tie of length at most three [7].

MAX SMTI is hard to approximate within a factor of $\frac{21}{19}(> 1.10)$ (unless P = NP) [9] and Yanagisawa [21] improves this bound to $\frac{33}{29}(> 1.13)$. He also shows that MAX SMTI cannot be approximated within $4/3$ under the assumption that the minimum vertex cover problem cannot be approximated within a factor of $2 - \epsilon$.

Manlove at al. [[17]] show that MAX SMTI is NP-complete, even if the ties are at the tails of the lists and on one side only, there is at most one tie per list, and each tie is of length 2.

The same results holds for a variant with minimum cardinality.

3.3 Optimal Solution

In the previous section we present that MAX SMTI is NP-hard which means that there is not any polynomial algorithm providing $P! = NP$. Anyway, we present one here two possible algorithms how to solve MAX SMTI the cardinality of which is not big.

Remeber that we can compute a stable marriage for an instance of SMTI by arbitrarily breaking all ties occuring in the instance. By another breaking ties we can get another stable marriage of different size. So, the first approach to achieve an optimal solution is a brute-force approach in which we compute all possible stable marriages and then choose the best one. The execution is very exhausted and very slow even for small instances.

Another approach is to use an integer programming introduced by Roth et al. [5]. We formulate an integer program as follow:

Maximize:

$$\sum_i \sum_j x_{i,j}$$

Subject to:

$$\forall w : \sum_i x_{i,w} \leq 1$$

$$\forall m : \sum_j x_{m,j} \leq 1$$

$$\forall (m,w) \in A : \sum_j x_{m,j} + \sum_i x_{i,w} - x_{m,w} \geq 1$$

$$\forall (m,w) \notin A = 0$$

$$\forall (m,w) \in \{0,1\}$$

where $x_{i,j}$ denotes if a pair m_i, w_j is in a stable marriage and A denotes a set of acceptable pairs. This approach is much more better than the previous one but its execution still takes a long time for bigger instances, e.g. instances of size more than 100.

Algorithm 3.1 OPTIMAL [5]

Input: an instance I of SMTI

Output: a stable matching M

```
1: formulate the instance as an integer program $IP(I)$
2: $x^* \leftarrow$ optimal $IP(I)$ solution
3: $M \leftarrow \{(m_i, w_j) \mid x^*_{i,j} = 1\}$
4: return $M$
```

Chapter 4

Approximation Algorithms for MAX SMTI

This section outlines some approximation algorithms for solving the Max SMTI problem which is NP-hard problem. As the following lemma says that there exists a trivial 2-approximation algorithm.

Lemma 4.1 ([17]). *For an arbitrary instance of SMTI, the size of the largest stable matching is at most twice the size of the smallest one.*

Notation 4.2. Let P be a minimazation problem and let T be the appropriate approximation algorithm. Let $opt(x)$ and $T(x)$ be the costs of optimal solution and a solution obtained by the algorithm T, respectively. We say that T is $r(N)$-approximation algorithm if for every x: $\frac{T(x)}{opt(x)} \leq r(N)$. If such a polynomial algortihm exists, then we say that P is approximable within $r(N)$. (for maximization problem: $\frac{opt(x)}{T(x)} \leq r(N)$)

4.1 RandBrk

Halldórsson et al. [8] give a randomized algorithm RANDBRK with expected approximation ratio 10/7 for a restricted case, where ties are only on women's lists, each woman writes at most one tie per list and each tie is of length 2. That also claim these restrictions, except for ties of length 2, can be removed whithout significantly increasing the approximation ratio. Algorithm 4.1 shows how RANDBRK works.

Algorithm 4.1 RANDBRK [8]

```
Input: a restricted instance $\hat{I}$ of SMTI
Output: a stable matching $M$
1: for all man $m$ who writes a tie $t$ do
2:     break the tie $t$ with equal probability        // $w_1$ precedes $w_2$ with probability $1/2$
3: end for
4: $I \leftarrow$ the resulting SMI instance
5: $M \leftarrow$ a stable matching given by G-S algorithm
```

4.2 ShiftBrk

Halldórsson et al. [9] introduce an approximation algorithm SHIFTBRK (Algorithm 4.2) with an approximation ratio of $\frac{2}{1+L^{-2}}$ for restricted cases when only women have ties and each tie is of length at most L. In case of $L = 2$ (both men and women are allowed to write ties) the approximation ratio is $\frac{13}{7}$.

This algorithm is based on iteratively shifting ties. Let $(p_1, p_2, \ldots, p_k)$ be a tie t and $t' = [p_1, p_2, \ldots, p_k]$ a list obtained by arbitrarily breaking t. We define an operation SHIFT(t') as follows:

$$\text{SHIFT}([p_1, p_2, \ldots, p_k]) = [p_2, \ ldots, p_k, p_1]$$

This operation represents a tie shifting. Now, we define another operation SHIFT$_p(I)$ where I is a SMTI instance and p is a man or a woman. The operation SHIFT$_p(I)$ returns new instance I' obtained from I by shifting all ties in p's preference list.

Algorithm 4.2 SHIFTBRK [9]

```
Input: a SMTI instance $I$
Output: a stable matching $M$
 1: arbitrarily break all ties in $\hat{I}$, let $I_{1,1}$ be the resulting SMI instance
 2: for all $i \in \{2, \ldots, L\}$ do
 3:     construct a SMI instance $I_{i,1} = Shift_w(I_{i-1,1})$
 4: end for
 5: for all $i \in \{1, \ldots, L\}$ do
 6:     for all $j \in \{2, \ldots, L\}$ do
 7:         construct a SMI instance $I_{i,j} = Shift_m(I_{i,j-1})$
 8:     end for
 9: end for
10: for all $i$ and $j$ do
11:     $M_{i,j} \leftarrow$ G-S($I_{i,j}$)
12: end for
13: return the largest matching among all $M_{i,j}$'s
```

4.3 IMY

Iwama, Miyzaki and Yamouchi [14] (IMY) present the first non-trivial approximation algorithm with fixed approximation ratio better than 2 for general SMTI instances. The algorithm with an approximation ratio of 1.875 is based on a local search and iteratively tries to find better stable matching.

In the algorithm we use some new notations: $BP_{s,m}(M)$ is a set of blocking pairs such that man is single and woman is matched, $BP_{-,m}(M)$ is a set of blocking pairs where woman is matched, notation M^X denotes the set of men who are married in M.

It uses two subroutines, INCREASE (Algorithm 4.5) and STABILIZE (Algorithm 4.4). The first subroutine takes a stable matching M as an input parameter. It is proved that if the size of M is less than $\frac{8}{15}OPT$ then INCREASE finds a matching of bigger size, otherwise it fails. Note that the resulting matching is not necessarily a stable matching, but it satisfies the following property Π about blocking pairs:

Π: If (m, w) is a blocking pair then either w or w is single.

When INCREASE returns such a matching then STABILIZE comes into play. As the name suggests, the subroutine makes a given matching stable without decreasing the size.

Algorithm 4.3 LOCALSEARCH [14]

```
Input: an instance $I$ of SMTI
Output: a stable matching $M$
1: $M \leftarrow$ an arbitrary stable matching for $I$
2: while true do
3:   $M' \leftarrow$ INCREASE($M$)
4:   if INCREASE returns an error then
5:     return $M$
6:   end if
7:   $M \leftarrow$ STABILIZE($M'$)
8: end while
```

Algorithm 4.4 STABILIZE [14]

```
Input: a matching $M_0$ with property $\psi$
Output: a stable matching $M$
1: while $BP_{s,m}(M_0) \neq \emptyset$ do
2:   select $(m, w) \in BP_{s,m}(M_0)$
3:   $w^* \leftarrow$ woman s.t. $(m, w^*) \in BP_{s,m}(M_0) \wedge \nexists (m, w') \in BP_{s,m}(M_0) : w' \succ_m w^*$
4:   $M_0 \leftarrow M_0 \setminus \{(M_0(w^*), w^*)\} \cup \{(m, w^*)\}$
5: end while
6: while $BP_{-,s}(M_0) \neq \emptyset$ do
7:   select $(m, w) \in BP_{-,s}(M_0)$
8:   $m^* \leftarrow$ man s.t. $(m^*, w) \in BP_{s,m}(M_0) \wedge \nexists (m', w) \in BP_{-,s}(M_0) : m' \succ_w m^*$
9:   if $m^*$ is matched in $M_0$ then
10:    $M_0 \leftarrow M_0 \setminus \{(m^*, M_0(m^*))\} \cup \{(m^*, w)\}$
11:  else
12:    $M_0 \leftarrow M_0 \cup \{(m^*, w)\}$
13:  end if
14: end while
```

Algorithm 4.5 INCREASE [14]

```
Input: a stable matching $M$
Output: a matching $M'$ with property $\Pi$
 1: construct a bipartite graph $G_1 = (U_1, V_1, E_1)$ where
            $U_1 \leftarrow M^X$
            $V_1 \leftarrow Y \setminus M^Y$
            $E_1 = \{(w, m) \subseteq U_1 \times V_1 \mid M(w) =_w m\}$
 2: $P \leftarrow$ a maximum matching in $G_1$
 3: $M' \leftarrow \{(m, w) \in M \mid w \in P^X\}$
 4: $M_1 \leftarrow M \setminus M' \cup P$
 5: $F \leftarrow X \setminus M_1^X$
 6: for all $m \in M'^Y$ do
 7:     $M_m \leftarrow M_1 \cup \{(m, w^*) \mid w^*$ is the woman whom $m$ prefers best within $F\}$
 8:                  // select an arbitrary woman if there are ties, $M_m$ is undefined if there is no such $w^*$
 9: end for
10: construct a bipartite graph $G_2 = (U_2, V_2, E_2)$ where
            $U_1 \leftarrow M'^X$
            $V_2 \leftarrow M_1^X$
            $E_2 \leftarrow \{(m, w) \mid M_m$ is defined $\wedge (m, w)$ blocks $M_m\}$
11: $Q \leftarrow$ men-propose G-S algorithm to $G_2$                // break ties arbitralily
12: $M'' \leftarrow \{(m, w) \in M_1 \mid w \in Q^X\}$
13: $M_2 \leftarrow M_1 \setminus M'' \cup Q$
14: for all $m \in M'^Y \setminus M_2^Y$ do
15:     $M^* \leftarrow M_2 \cup \{(m, w^*) \mid w^*$ is the woman whom $m$ prefers best within $F\}$
16:     if $M^*$ satisfies $\Psi$ then
17:         return $M^*$
18:     else
19:         stop
20:     end if
21: end for
22: do the same operations as lines 1 through 21 by exchanging the role of men and
    women
23: return an error
```

4.4 SSMTIApprox

Irving and Manlove [12, 11] describe a $\frac{5}{3}$-approximation algorithm[1] for SSMTI instances. We revise that a SSMTI instance is a special SMTI instance where only women are allowed to write at most one tie at the end. SSMTIAPPROX consists of 3 phases.

Notation 4.3. Let w be a woman, let t be a tie at the head of w's list and let m be a man in a tie t. If t contains some other man m_1 besides m then we say that m is an *tied man* and w is an *tied woman*. Let m' be a man and let w' be a woman. We say that a pair (m', w') is *Phase 1 - acceptable* if (m', w') is an acceptable pair after all deletions during execution Phase 1.

Phase 1

It is a variant of G-S algorithm where women make proposals. The difference is in pausing a woman when she has some untied man at the head of her list. Note that moving on preference list is here processed by deleting proposed men, e.g. next man is a man at the head of the list.

Algorithm 4.6 SSMTIAPPROX Phase 1 [9]

```
 1: set all persons to be free
 2: while there exists a free woman $w$ with non-empty list and has an untied man $m$
    ath the head of her list do
 3:   $w$ proposes and becomes married with $m$
 4:   for all successor $w'$ of $w$ on $m$'s list do
 5:     if $w'$ is married with $m$ then
 6:       break the marriage                                   // $w'$ is now free
 7:     end if
 8:     delete the pair $(m, w)$ from preference lists
 9:   end for
10: end while
11: go to Phase 2
```

Phase 2

In this phase we find a maximum matching between tied men and tied women. Then, for each pair (m, w) in this maximum matching we promote m ahead in w's list.

Phase 3

In this phase we completely break the remaining ties to get a SMI instance and subsequently we apply standard G-S algorithm where men make propsals. SMI instance.

1. [11] fixes an incorrect approximation ratio of $\frac{8}{5}$ to the correct one of $\frac{5}{3}$

Algorithm 4.7 SSMTIAPPROX Phase 2 [9]

```
1: $U \leftarrow$ tied men
2: $V \leftarrow$ tied women
3: $E \leftarrow \{(m, w) \in U \times V \mid (m, w)$ is a Phase 1 accaptable pair$\}$
4: construct a bipartite graph $G = (V, E)$
5: $K \leftarrow$ a maximum cardinality matching in $G$
6: if $K = \emptyset$ then
7:     go to Phase 3
8: end if
9: for all $(m, w) \in K$ do
10:     promote $m$ from the tie to the head o $w$'s list
11: end for
12: re-activate the proposal sequence
13: go to Phase 1
```

Algorithm 4.8 SSMTIAPPROX Phase 3 [9]

```
1: for all woman $w$ do
2:     breal the tie (if any) in $w$'s list, placing tied men ahead of the untied men
3: end for
4: set all men as free
5: apply G-S algorithm where men make proposals
6: return resulting stable matching
```

4.5 Kiraly

Kiraly [16] gives a better and simpler approximation algorithms than the previous ones. The first one, GSA 1 (Algorithm 4.9), is a linear $\frac{3}{2}$ - approximation algorithm for SMTI instances where only women are allowed to write ties. The latter one, GSA 2 (Algorithm 4.10) is designed for general SMTI instances with an approximation ratio of $\frac{5}{3}$. The algorithm runs in time $O(|E|)$.

GSA 1

Algorithm 4.9 uses a second chance approach which means the following. Each man makes proposals exactly as in G-S algorithm. When he is out of choices, he gets an extra score ϵ - the second chance. He is re-activated and starts proposing from the beginning of his preference list with an important difference. When he makes a proposal to a woman, this woman uses pri' to decide his rejection (this is what Király calls reduced men-proposal G-S). Since $\epsilon = 1/2$ the woman decides with different dicision (than for the first time) only when she finds the man in a tie. Let SM denote a set of single men and let Π_o denote a set od men with zero extra score.

Algorithm 4.9 GSA1 [16]

Input: an instance I of SMTI where men have strictly ordered lists
Output: a stable matching M

```
1: run GS
2: for all $m \in U$ do
3:     $\pi(m) \leftarrow 0$
4: end for
5: while $SM \cap \Pi_0 \neq \emptyset$ do
6:     for all $m \in SM \cap \Pi_0$ do
7:         $\pi(m) \leftarrow \epsilon$
8:         re-activate $m$
9:     end for
10:    run rmGS                                  // reduced men proposal G-S
11: end while
```

GSA 2

Algortihm 4.10 is designed for general SMTI instances. At first,we run the algorithm GSA 1 (Algorithm 4.9). During this execution some men might get extra score. When GSA 1 finishes, we break women's lists taken into account men's extra score, e.g. men with extra score ϵ are promoted ahead of men with zero extra score. All other ties are broken arbitrarily. Next, we change the roles of men and women, which means that women make proposals. This time only women gets extra score.

We define algorithm $rwGS$ (reduced women-proposed G-S) similarly to $rmGS$. But here is a major difference: Ia a woman w with $\pi(w) = 0$ is rejected by her actual partner at any time during the process then she gets $\pi(w) = \epsilon/2$ extra score, activates herself and starts making proposals from the beginning of her strict list. If some women with extra score less than ϵ we increase their extra scores to ϵ and re-activate them. Let SW denote single women and let Π denote a set of women with extra scores less than ϵ.

4.6 McDermid

In [19] McDermid presents a $\frac{3}{2}$ approximation algorithm for general MAX SMTI problem. He uses ideas from Kiraly 4.5 and SSMTIApprox 4.4. The algorithm runs in time $O(n^{\frac{3}{2}}L)$ where n is the sum of men and women and L is the sum of all preference lists.

Notation 4.4. Let m be a man, we say that m is $exhausted$ if he is out of choices. If m is re-activated to start making proposals from the beginning of his list then we say that m

Algorithm 4.10 GSA 2 [16]

```
Input: an instance $I$ of SMTI (general case)
Output: a stable matching $M$
 1: $I' \leftarrow$ new instance obtained from $I$ by arbitrarily breaking ties in men's lists
 2: run GSA1$(I')$
 3: for all $w \in V$ do
 4:     $\pi(w) \leftarrow 0$
 5: end for
 6: while $SW \cap \Pi \neq \emptyset$ do
 7:     for all $m \in SW \cap \Pi$ do
 8:         $\pi(w) \leftarrow \epsilon$
 9:         re-activate $w$
10:     end for
11:     run rwGS                                  // reduced women proposal GS
12: end while
```

is *promoted*. Let t be a current tie on m' list. If there are at least two unmatched women in t then m pauses his proposing (like in Section 4.4) and is set as *stalled*.

Theorem 4.5 (Gallai-Edmonds decomposition). *Let $G = (u \cup V, E)$ be a bipartite graph and M be a maximum cardinality matchong for G. Let $\mathcal{E}, \mathcal{O}$ and $\mathcal{U}$ be the set of even, odd and unreachable vertices as defined above with respect to G and M. Then*

1. *$\mathcal{E}, \mathcal{O}$ and $\mathcal{U}$ are pairwise disjoint. Every maximum matching for G partitions the vertex set of G into the same sets of even, odd, and unreachable vertices.*
2. *In any maximum-cardinality matching of G, every vertex in $\mathcal{O}$ is matched with some vertex in $\mathcal{E}$, and every vertex in $\mathcal{U}$ is matched with another vertex in $\mathcal{U}$. The size of maximum-cardinality matching is $|\mathcal{O}| + |\mathcal{U}|/2$.*
3. *There is no edge in G connecting a vertex in $\mathcal{E}$ with a vertex in $\mathcal{U}$*

The McDermid algorithm consists of 3 phases:

Phase1

If a man m is out of choices in his preference list, m is set to be *exhausted*. If m is exhausted for the first time, he is set to be *promoted* and he is re-activated (like in Section 4.5). If m has in his current tie at least two unmatched women, he is set to be *stalled* and pauses his proposing (like in Section 4.4).

Phase2

The goal of this phase is to attempt to match a certain subset of stalled men to women appeared in men's current ties, e.g. a man can be matched only to woman who appears

Algorithm 4.11 MCDERMID: PHASE 1 [19]

Input: an instance I of SMTI (general case)

Output: a stable matching M

```
1:  $M \leftarrow \emptyset$
2:  set all men to be unmathced, unpromoted, unexhausted and unstalled
3:  **Phase 1:**
4:  **while** $\exists m$ : $m$ is unmatched **and** unstalled **and** ($m$ unpromoted **or** unexhausted) **do**
5:     **if** $m$ is exhausted **then**
6:        promote $m$ and set $m$ to be unehausted
7:        reactivate $m$ so that he begins proposing again from the start of his list
8:     **end if**
9:     $t \leftarrow$ $m'$s current tie
10:    **if** $|t| \geq 2$ **then**
11:       **if** $t$ contains exactly one unmatched woman $w$ **then**
12:          promote $W$ ahead of $t$
13:       **else if** $t$ contains no unmatched woman **then**
14:          break $t$ arbitrarily
15:       **else**
16:          set $m$ to be stalled
17:       **end if**
18:    **else**
19:       $w \leftarrow$ only mwoman in $t$                          // $m$ proposes to$w$
20:       **if** $w$ is unmatched **then**
21:          $M \leftarrow M \cup (m, w)$                          // $w$ accepts $m$
22:          unstall the appropriate men, if any
23:       **else if** $w$ prefers $m$ to her partner $m'$ **then**
24:          $M \leftarrow M \cup \{(m, w)\} \setminus \{(m', w)\}$        // $w$ rejects $m'$ and accepts $m$
25:          set $m'$ to be exhausted if $w$ is the last woman on his list
26:       **else**
27:          set $m$ to be exhausted if $w$ is the last woman on his list
28:       **end if**
29:    **end if**
30: **end while**
31: **if** the set $S$ of stalled men is empty **then**
32:    **return** $M$
33: **else**
34:    **go to** Phase 2
35: **end if**
```

in his current tie.

Algorithm 4.12 MCDERMID: PHASE 2 [19]

```
1: construct a bipartite graph $G = (U \cup V, E)$ where
       $U \leftarrow$ stalled men $S$
       $V \leftarrow$ free women appearing in the current tie of $m \in S$
       $E = \{(m, w) \in U \times V \mid w$ appears in $m$'s current tie $\}$
2: $N \leftarrow$ maximum cardinality matching in $G$
3: identify the sets $\mathcal{E}, \mathcal{O}$ and $\mathcal{U}$          // Gallai-Edmonds decomposition
4: $N' \leftarrow$ subset of $N$ obtained by removing all pairs $\{(m, w) \mid m \in \mathcal{O}, w \in \mathcal{E}\}$
5: if $N' = \emptyset$ then
6:     go to Phase 3
7: else
8:     for $(m, w) \in N'$ do
9:         promote $w$ ahead of $m$'s current tie          // $m$ will propose to $w$ in Phase 1
10:        $M \leftarrow M \cup (m, w)$
11:        set $m$ to be unstalled
12:    end for
13:    unstall all men in $\mathcal{U}$ who are unmatched in $N$
14:    go to Phase 1
15: end if
```

Phase3

Phase 3 takes the current matching M and the maximum matching N computed in Phase 2. All pairs in N are added to M and the algorithm return the stable matching M.

Algorithm 4.13 MCDERMID: PHASE 3 [19]

```
1: for $(m, w) \in N$ do
2:     $M \leftarrow M \cup (m, w)$          // $m$ proposes to $w$
3: end for
4: return $M$
```

4.7 Paluch

Paluch [20] gives another $\frac{3}{2}$-approximation algorithm for general MAX SMTI problem that runs in $O(|E|)$ time where $|E|$ is the number of edges. The algorithm consists of two procedures:

1. **GS Modified** (Algorithm 4.14)
   It is a modified G-S algorithm

2. **GS Improve** (Algorithm 4.15)
   It improves the existing stable matching

Definition 4.6. Let (m, w) be a pair in some matching M and let t be a tie in m's list such that $w \in t$. If there exists a free woman $w_1 \in t$ then we say that m is *unstable* and that w_1 is a *satellite* of m also a woman w is *unstable*. Let $e = (m', w')$ be an edge such that both a man m' and a woman w' are free and there is at least one free woman w'_1 such that w' and w'_1 are equally good for m, then e is called *special*. In other words it means that a free woman w'_1 appears in the same m's tie as w'.

Definition 4.7. Let M be a matching and let P be a path among men and women. We say that P is an *alternating* path if its edges are alternating from M and ones not in M. Consider an alternating path $P = (w, m_1, w_w, m)$ where w and m are both unmatched and (m_1, w_1) is not a blocking pair for matching $M' = \{(w, m_1), (w_1, m)\}$. Such a path is called *dangerous* path. Note that if P is a dangerous path then m_1 is indifferent between w and w_1 (we denote $w, m1$ as an *mequal* edge) or w_2 is indifferent between m and m_1 (we denote $w, m1$ as *fequal* edge) or both. Thus a dangerous path must contain one or two equal edges. If a dangerous path P contains a mequal edge then we say that P is a *masculine* dangerous path. If a dangerous path P contains a fequal edge then we say that P is a *feminine* dangerous path. Figure 4.1 illustrates both types of dangerous paths. A mna is said to be a *spoiling* man if he belongs to a blocking pair or if he is free and belongs to a masculine dangerous path.

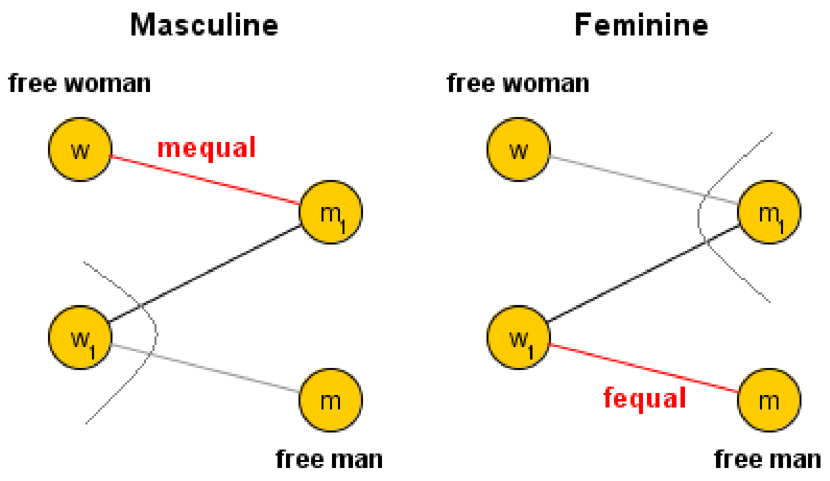

Figure 4.1: Masculine and feminine dangerous paths.

GS Modified

GS MODIFIED (Algorithm 4.14) is very to similar to G-S algorithm with one difference - a detection of masculine dangerous paths (line 5). We always move forward on a list L_m except one case - when a man m processes a special edge (m, w). In this case we do not move forward on m's list and in the next time in the algorithm we consider w again. It means that m pauses his proposing. GS MODIFIED computes a stable matching and there are no masculine dangerous paths.

Algorithm 4.14 GS MODIFIED [20]

```
Input: an instance $I$ of SMTI (general case)
Output: a stable matching $M$
1: while there is a free man $m$ with a nonempty list do
2:     $w \leftarrow$ next woman on $m$'s list
3:     if $m$ is better for $w$ than $M(w)$ or $w$ is free then
4:         $M \leftarrow M \cup (m, w) \setminus (w, M(w))$
5:     else if $w$ is unstable and there is no free woman $w'$ who is equally good for $m$ as
       $w$ then
6:         $M \leftarrow M \cup \{(m, w), (M(w), w')\} \setminus (w, M(w))$        // $w'$ is satellite of $M(w)$
7:     end if
8: end while
```

GS Improve

GS IMPROVE (Algorithm 4.15) eliminates dangerous paths where eliminating blocking pairs and mascilune dangerous path has a higher priority than eliminating feminine dangerous paths. The algorithm looks for a free man who is a spoling man or if he is a part of a feminine dangerous path. Note that at each iteration there is at most one spoling man. The algorithm uses an auxiliary preference list L'_m for each man m. L'_m is initialized to be empty and only men who are a part of a feminine dangerous path are added into it.

4.8 GSA-LP

A new approach using linear programming is given by Iwama et al [15]for SMTI instances where only women have ties. Their algorithm (Algorithm 4.16) is based on Kiraly's algorithm GSA-1 in a sense of getting extra scores. The approximation ratio is improved from $\frac{3}{2}$ to $\frac{25}{17}$ (< 1.4706). At first, we formulate an instance as an integer program as described in Section XX. We solve its LP relaxation to get an optimal solution which assigns a possitive real number po each pair. These numbers are used in a variable $f(m)$ which is similar to Kiraly's extra scores. A variable $p(m)$ determines a position on m's list.

Algorithm 4.15 GS IMPROVE [20]

Input: an instance I of SMTI (general case)
Output: a stable matching M

```
1: while there are free spoiling men or feminine dangerous paths do
2:   if there is a free spoiling man $m$ then
3:     $w \leftarrow$ next woman on $m'$s list $L_m$
4:     if $(m, w)$ is blocking then
5:       $M \leftarrow M \cup (m, w) \setminus (w, M(w))$
6:     else if $w$ is unstable and there is no free woman $w'$ who is equally for $m$ as $w$
       then
7:       $M \leftarrow M \cup \{(m, w), (M(w), w')\} \setminus (w, M(w))$        // $w'$ is a stellite of $M(w)$
8:     else if $(m, w)$ is fequal then
9:       add $w$ to the end of list $L'_m$
10:    end if
11:  else if there is a feminine dangerous path $P$ containing free man $m$ then
12:    $w \leftarrow$ next woman on $m'$ lists $L'_m \cup L_m$        // we scan at first $L'_m$
13:    if $(m, w)$ is fequal then
14:      $M \leftarrow M \cup (m, w) \setminus (w, M(w))$    // at this step $M(w)$ is the only one possible spoling man
15:    end if
16:  end if
17: end while
```

Algorithm 4.16 GSA -LP [15]

Input: an SMTI instance I where only women have ties
Output: a stable matching M

```
1: formulate the given instance $I$ as in integer program $IP(I)$
2: solve its $LP$ relaxation to obtain an optimal solution $x^*_{i,j}$
3: $M \leftarrow \emptyset$
4: for all man $m$ do
5:    $f(m) \leftarrow 0$
6:    $p(m) \leftarrow 1$
7: end for
8: while there exists a free man $m$ with $f(m) \leq 3$ do
9:    if $p(m)$ is larger than the length of $m$'s preference list then
10:       $f(m) \leftarrow f(m) + 1$
11:       $p(m) \leftarrow 1$
12:    else
13:       $w \leftarrow p(m)$-th woman on $m$'s preference list
14:       if $m$ has not proposed to $w$ yet then
15:          $f(m) \leftarrow f(m) + x^*_{m,w}$
16:          let $m$ propose to $w$
17:          $p(m) \leftarrow 1$
18:       else
19:          let $m$ propose to $w$
20:          $p(m) \leftarrow p(m) + 1$
21:       end if
22:    end if
23: end while
24: return $M$
```

Chapter 5

Implementation Remarks

In this chapter we describe details how to implements algorithms described in Chapter 4. For implementing these algorithms we choose a Java programming language due to its good simplicity and readability. But there is one exception we have to emphasize. Algorithms that use a linear programming we use an extrenal program `lp_solve` which is written in C.

We do not focus on implemantion details such as explainig source code, which is available in an attached CD. The CD also contains a JavaDoc documentation. All implemented algortihms are available at
via an applet the use of which is described in the last section of this chapter.

5.1 SMTI Instance Implementation

In Section 2.2 we describe various instances that all of our algorithms use. The hierarchy if all implemented unstances is depicted on Figure 5.1.

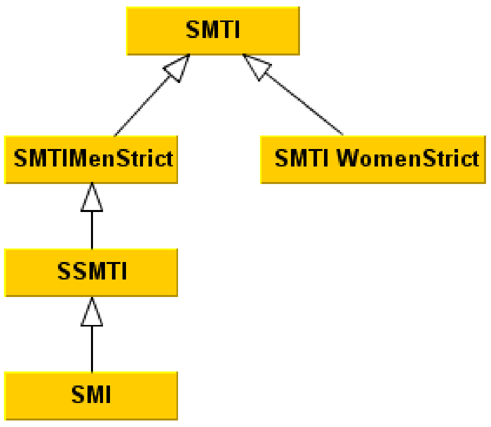

Figure 5.1: Hierarchy of instances

All instances extending `SMTI` only override the methods used for creating an instance. Using these methods they implement appropriate restrictions. Each instance can be initialized from a string or a file satisfying the form depicted on Figure 5.4. The first part refers to men's preference lists and the second part referes to women's preference lists where the most preffered person is placed leftmost (after colon). Note that this two parts must be separated by a blank line.

At fisrt, we describe how we represent a SMTI instance and how are some methods implemented. Since we know the cardinality in advance, we bound our implementation to this cardinality, say N. Remind the properties and requirements for SMTI instance:

- there is a set of men and a set of women, both of cardinality N
- each person has assigned a preference list of cardinality at most N
- deletion from a preference list must be done in constant time
- preference lists might contain ties
- breaking these ties must be in a reasonable time, e.g. in time $O(T)$ where T is the length of the tie

For representing a person we create an abstract class `Person` with all methods that are associated with persons. Since all algorithms works with men and women, we also create appropriate classes `Man` and `Woman` . These classes inherit from `Person` and do not implement some special methods other than `Person` , they are implemented especially for the reason of better readablity. Each person must have a preference list with the required properties described above. We implement the preference list using a data structure called *bounded linked list* which is described bellow.

5.1.1 Bounded Linked List

Bounded linked list is variant of well konwn data structre linked list with some differences. For an initialization we need a bound - the cardinality. This is a desired restriction for our purpose. During the initialization a new hash table of length N is created, as well. This hash table stores pointers to elements which allows us to delete an element from the preference list in constant time. Bounded Linked list is depicted on Figure 5.2.

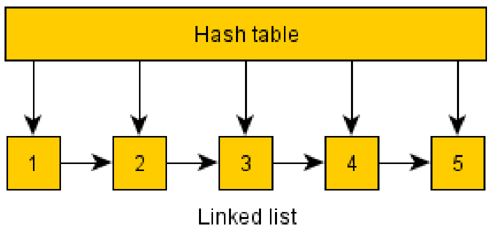

Figure 5.2: Bounded Linked List

5.1.2 Preference lists and ties

We represent a preference list as a list of ties that contains persons. An implemantation of a tie stores a set of persons, an owner and an owner's choice number. The owner and his choice number uniquely determine a tie. Note that it is allowed that a tie contains only one person.

The preference list (Figure 5.3) extends bounded linked list where elements are represented as ties and the hash table contains person pointers to these ties. Such an implementation allows us to add or delete a person in constant time. Furthermore, it allows us to promote a person ahead of the tie in constan time:

1. find the tie that contains the person, let c be the appropriate choice number
2. remove the person from that tie and increment the choice number
3. create new tie containing the person with choice number $= c$

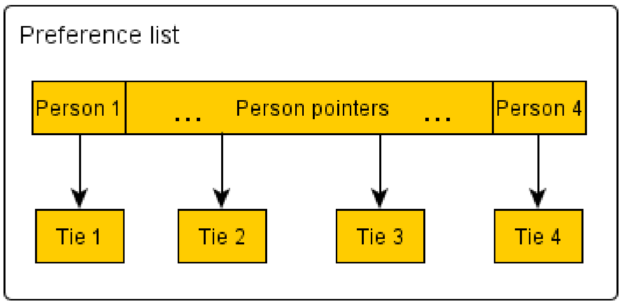

Figure 5.3: Preference list with ties

5.2 Applet

As a part of this thesis is an applet that allows us to use all mentioned algorithms via web interface. This applet is available at:
`http://www.fi.muni.cz/~xpodhr/stablemarriage/stablemarriage.html`

```
1:  3   4
2:  3   (1   4)
3:  3   4    1   2
4:  1   3

1:  4   3   1
2:  (1  2)
3:  (1  4   2   3)
4:  4
```

Figure 5.4: String form for initializing an instance

Chapter 6

Algorithms Comparison

In this chapter we deals with practical comparisons of all algorithms described in Chapter 4. For this purpose we generate over 10000 instances of SMTI considering properties such as

- two main restrictions (only women have ties, each woman writes at most one tie at the end)
- the length of preference lists
- the number and the length of ties

All these instances are also available on an attached CD/DVD.

In our coparisons we use 3 basic classes of instance of SMTI. The list of classes with usable algorithms is in the following:

1. **SMTI General**
   restriction: without any restriction
   usable algorithms: RandBrk, ShiftBrk, IMY, Kiraly2, McDermid, Paluch, Optimal
2. **SMTI Men strict**
   restriction: all men have strictly ordered lists
   usable algorithms: Kiraly1, GSA-LP plus the ones above
3. **SMTI Special**
   restriction: all men have strictly ordered lists and each woman writes at most one tie at the end
   usable algorithms: SSMTIApprox plus the ones above

There are two important factors that we take into account in our comparisons, accuracy and duration. At first, we have a look at behaviour of algorithms on relatively small instances. It allows us to imagine what kind of algorithms can be used in real world. For some of such instances wa are able to compute an optimal solution. Primarily, we consider approximation results. Then, we take into account a duration and execute algorithms on large instances.

For each class of instance listed above we take into account the length of ties in our comparisons. In each case we measure accuracy and duration (in miliseconds) that is depicted on two bar charts. These charts show average results among studied class of instances. Each chart contains a label that indicates the number and what kind of instances is considered. Note that there is one exception about showing duration in cases when GSA-LP or Optimal algorithms are used. These two algorithms use linear (or integer) programming and they are implemented using an external program `lp_solve` [2].

6.1 Approximation results on small instances

In this section we compare algorithms on small instances, more precisely on instances the cardinality of which is 100. We focus especially on accuracy (approximation ratio) property. This study also gives us a notion about durations. As Figure 2.3 shows, an approximation ratio does not matter of the cardinality of instance.

SMTI General

First of all, we consider SMTI instances without any restrictions. Since ShiftBrk algorithm has time complexity in $O(N^2L^2)$, where N is the cardinality of instance and$L$ is the length of the longest tie, we first randomly choose 100 instances from our generated instances and execute all mentioned algorithms on them. This gives us a notion how the length of the longest tie affects a duration. As we can see at Figure 6.1 the slowest algorithm is ShiftBrk. In our next comparisons we must take into consideration the legth length of the longest tie, so we make 2 comparisons for each class of instance. The first one is for instances where the length of each tie is less than 10. The second one does not consider the length of ties, but the comparing is without ShiftBrk.

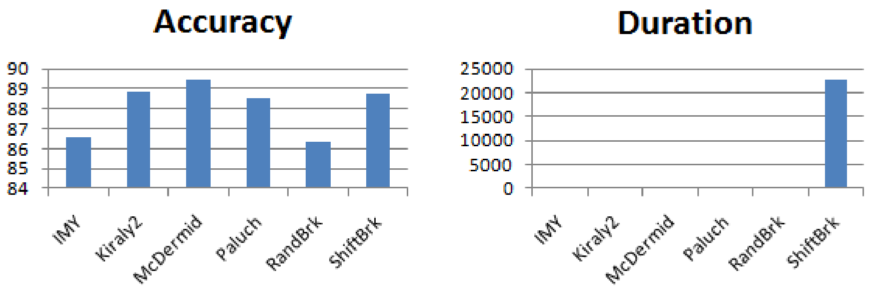

Figure 6.1: Random 100 instances of SMTI of size 100.

Figure 6.2 shows results for instances where all ties are smaller than 10. We see that a duration of ShiftBrk is much more better, but it is still very heigh. That is why we omit ShiftBrk when long ties come into play.

Now, we omit ShiftBrk algorithm and execute the other algorithms on SMTI instances without any restrictions (Figure 6.3). There are 10000 of such instances. For

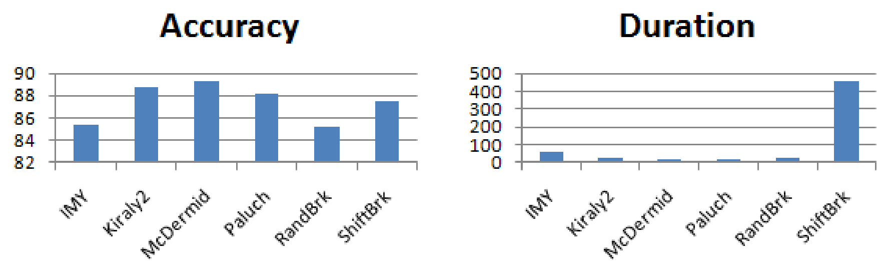

Figure 6.2: 1479 instances of SMTI where the length of each tie is less than 10.

information, executing ShiftBrk on such an instance takes about 1 minute. From this comparison we can observe that a duration of other algorithms seems to be independent on the length of the longest tie.

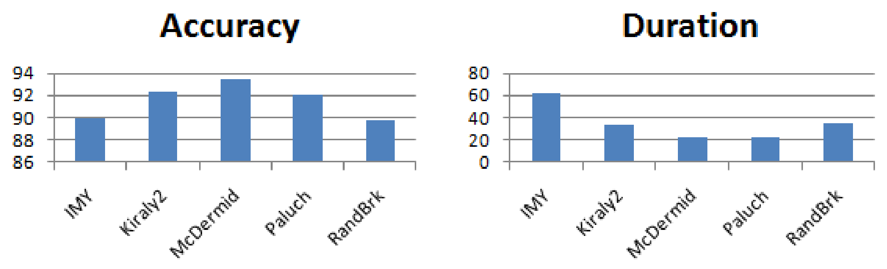

Figure 6.3: 10000 instances of SMTI.

To compare accuracy of approximation algorithms we create instances with optimal stable marriages of size 100. We do this by generating SM instances, computing stable marriage by GS algorithm, deleting pairs not contained in the stable marriage and finally by arbitrarily inserting ties into this instance. Results for our "optimal" instances are depicted on Figure 6.4. Here, we observe that the best results are given by algorithms Kiraly2, McDermid and Paluch. A duration of these algorithms is very good, as well.

Next, we try to execute Optimal algorithm (Algorithm 3.1) on SMTI instances where all tie are of length less than 10. Note that a duration of Optimal algorithm might be too high, even for such small instances. That is why we must restrict execution on some reasonable time, e.g. 100 seconds. Using this approach we success in 206 cases, results are captured on Figure 6.5. We can see that the results are not so bad. The worst results are given again by IMY and RandBrk.

The same approach we use for all general SMTI instances. Note that in this case we omit ShiftBrk because of its time complexity. Using Optimal algorithm we find 356 optimal solutions. We observe that the best algorithm is McDermid due to its very good accuracy and duration as well.

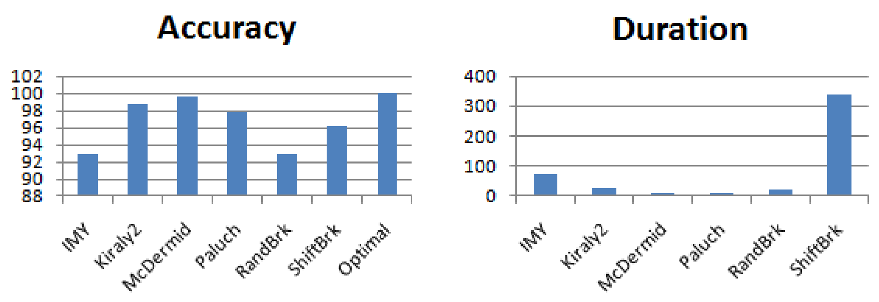

Figure 6.4: 768 ”optimal” instances of SMTI.

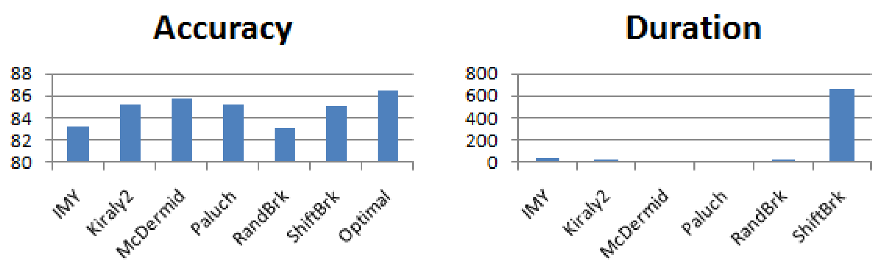

Figure 6.5: 206 instances of SMTI with optimal solutions where the length each tie is less than 10.

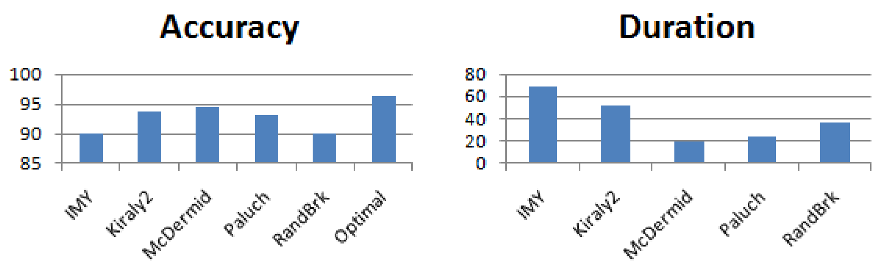

Figure 6.6: 1642 instances of SMTI.

SMTI Men Strict

Now, we consider SMTI instances where only women have ties in their preference lists. For this restriction of SMTI insatances we compare algorithms considered above, moreover, we can use algorithms Kiraly1 and GSA-LP. Note that our implementation of GSA-LP uses external program lp_solve [2], so we do not compare its duration.

Again, we start comparing with instances that have the length of each tie smaller than 10 (because of ShiftBrk). Figure 6.7 shows that the algorithms have very high accuracy due to our restriction. We can see that ShiftBrk has the best accurancy but also a very high duration which is undesirable.

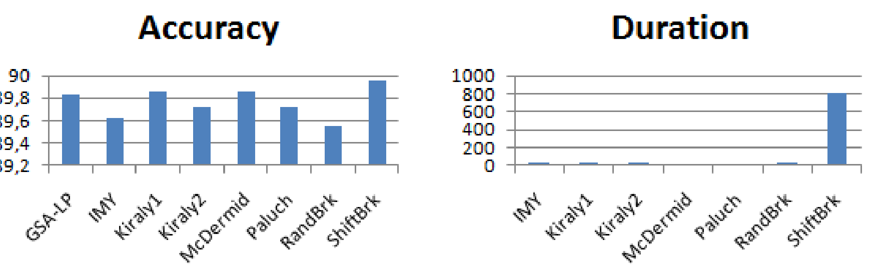

Figure 6.7: 317 instance of SMTI where only women have ties and the length of the longest tie is less than 10.

We omit ShiftBrk algorithm and apply the algorithms on all SMTI Men Strict instances. Figure 6.8 shows that new algorithm GSA-LP gives very good results as well as Kiraly1. Note that Kiraly2 gives worse results than Kiraly1.This is because Kiraly1 is designed for such instances.

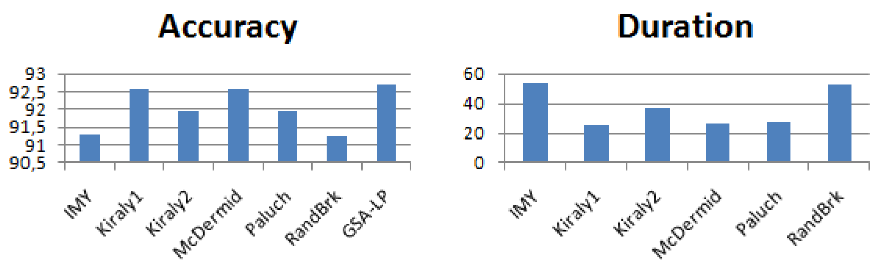

Figure 6.8: 1900 instances of SMTI where only women have ties.

Now, we compare our algorithms with optimal solutions for instances where all ties are smaller than 10. From Figure 6.9 we can observe that the results are very close to optimal. Again, algorithms IMY and RandBrk give the worst results.

Finally, we take all instances of SMTI Men Strict. Figure 6.10 reveals one interesting thing. The algorithm GSA-LP, which has the best approximation ratio, does not give

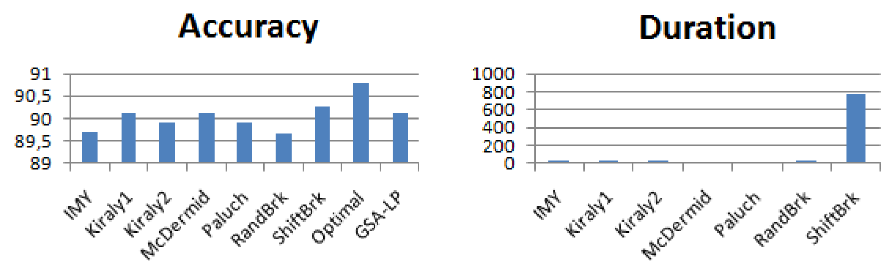

Figure 6.9: 149 instances of SMTI Men Strict where the length of each tie is less than 10.

better results than other approximation with worse approximation ratio.

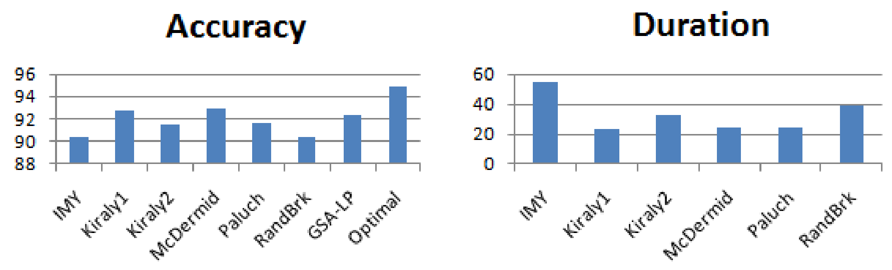

Figure 6.10: 1266 instances of SMTI Men Strict.

SMTI Special

The last class of SMTI instances is the most restricted variant of SMTI. In this case only women are allowed to write a tie, moreover, each woman can writes at most one tie at the end of her preference list. This variant are also used in practical applications.

We compare algorithms in the same way as in the two previous cases. In this case we can use new algorithm SSMTIApprox which is designed just for instances of SSMTI. In the first two comparisons (Figures 6.11, 6.12) the algorithm SSMTIApprox gives very good results and its duration is good as well.

In the next two comparisons SSMTIApprox gives a little bit worse results but but still is have one of the best results.

Haldorson, et al. [8] present an interesting SMTI instance where only women have ties and all ties of length at most L . For such an instance the approximation ratio of RandBrk is at least $O(\log L/L)$. This instance is depicted on Figure 6.15. We use this instance in our comparison for $L = 50$. On Figure 6.16 shows interesting situation where some algorithms find an optimal solution and the other ones find very poor stable marriage.

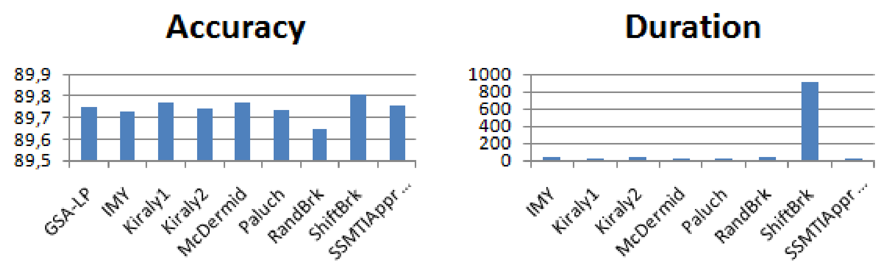

Figure 6.11: 194 instances of SSMTI where the leght of each tie is less than 10.

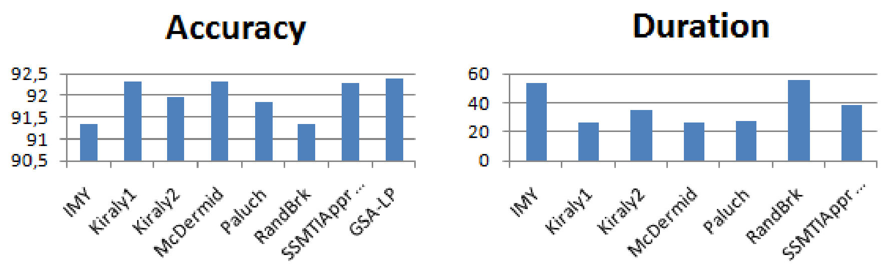

Figure 6.12: 1499 instances of SSMTI.

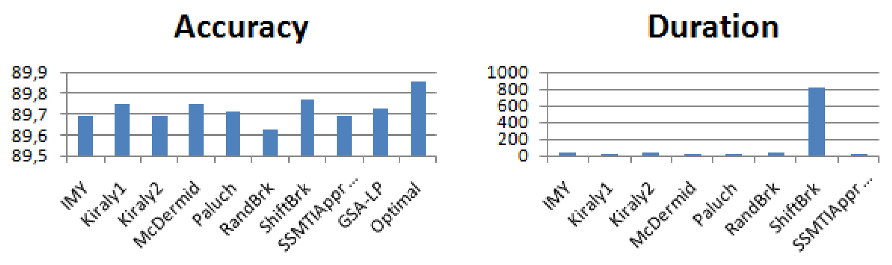

Figure 6.13: 48 instances of SSMTI where the length of each tie is less than 10.

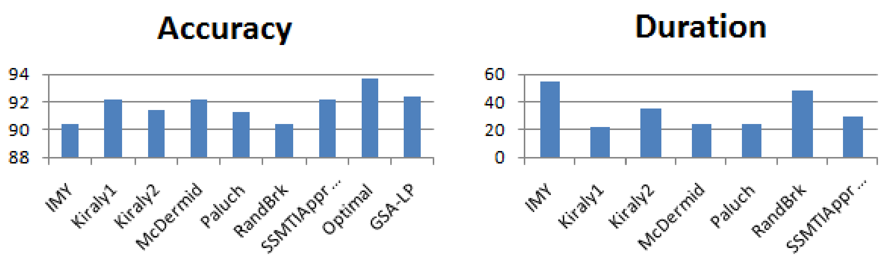

Figure 6.14: 640 instances of SSMTI.

$$
\begin{array}{llll}
\text{Men's list} & & \text{Women's list} & \\
m_1: & (w_1\, w'_1\, \ldots\, w'_{l-1}) & w_1: & m_1 \\
\vdots & \vdots & \vdots & \vdots \\
m_{l-1}: & (w_{l-1}\, w'_1\, \ldots\, w'_{l-1}) & w_{l-1} & m_{l-1} \\
m'_1: & w'_1 & w'_1: & m_1\, \ldots\, m_{l-1}\, m'_1 \\
\vdots & \vdots & \vdots & \vdots \\
m'_{l-1}: & w'_{l-1} & w'_{l-1}: & m_1\, \ldots\, m_{l-1}\, m'_{l-1}
\end{array}
$$

Figure 6.15: Interesting instance

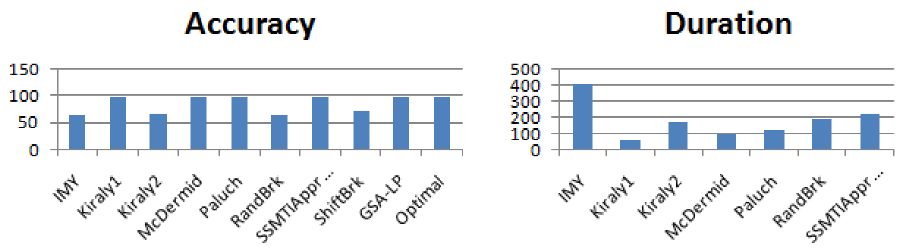

Figure 6.16: SMTI instance depicted on Figure 6.15 with $L = 50$

Observations

We can conclude that the best approximation algorithm according to our comparisons is the algorithm McDermid. In all our tests it gives the best results and it has the one of the lowest duration among all approximation algorithms.

Next algorithms with high accuracy are Paluch and Kiraly's algorithms where Kiraly1 are better than Kiraly2 for instances of SMTI Men Strict.

6.2 Approximation results on large instances

The previous section shows us which algorithms have high accuracy. It also gives us some notion about durations of the algorithms. In this section we focus especially on durations of the algorithms. We omit ShiftBrk algorithm since its duration is too high even it is applied on small instances.

SMTI General

At first, we compare algorithms on general SMTI instances of cardinality 1000. Figure 6.17 indicates that McDermid, Paluch and Kiraly2 give the best results. Note the duration of IMY which is extremely large.

In the next comparisons we are interested especially in algorithms McDermid and Paluch. We omit Kiraly2 because it has high duration. Since we are interested in dura-

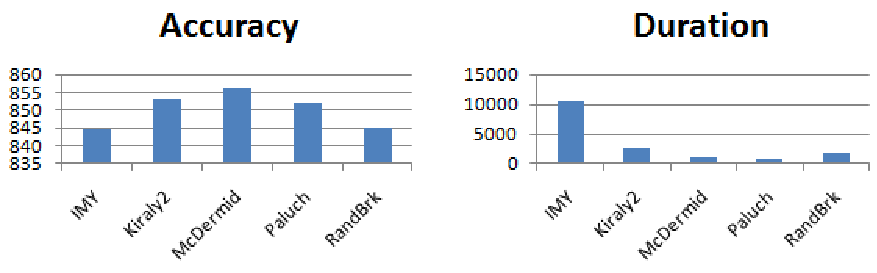

Figure 6.17: 300 instances of SMTI of cardinality 1000.

tions, we also omit RandBrk. So, in the next comparisons we compare only algorithms McDermid and Paluch. Figures 6.18 and 6.19 show that McDermid has better results than Paluch. Note that on small instances Paluch runs slower than McDermid, but on bigger instances it runs faster (Figure 6.19).

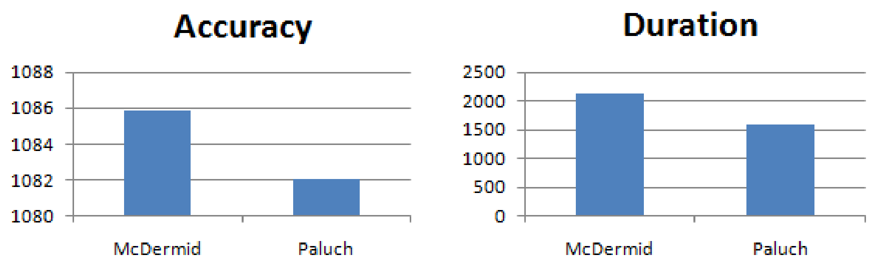

Figure 6.18: 200 instances of SMTI of cardinality 2000.

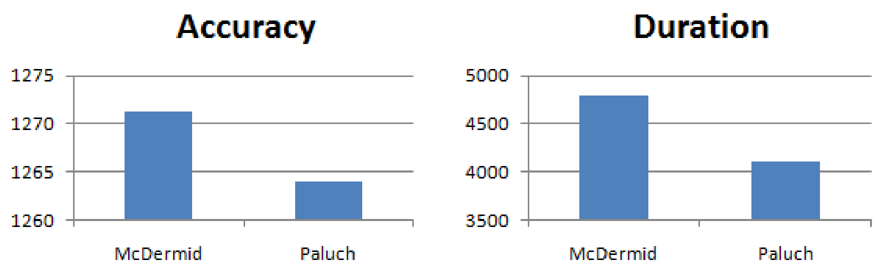

Figure 6.19: 3 instances of SMTIof cardinality 3000.

SMTI Men Strict

For the class SMTI Men Strict we compare only three algorithms Kiraly1, McDermid and Paluch. On Figure 6.20 we can see that Kiraly and McDermid return the same results. This is because McDermid algorithm comes from Kiraly1. On ther other side we can observe that McDermid runs faster than Kiraly1.

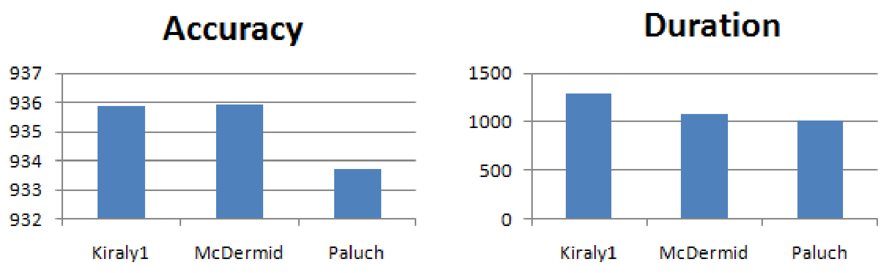

Figure 6.20: 200 instances of SMTI Men Strict of cardinality 1000.

SMTI Special

In this case of instances we put into our comparing SSMTIApprox algorithm. As we can see on Figure 6.21 the algorithm runs much more slower than the other ones.

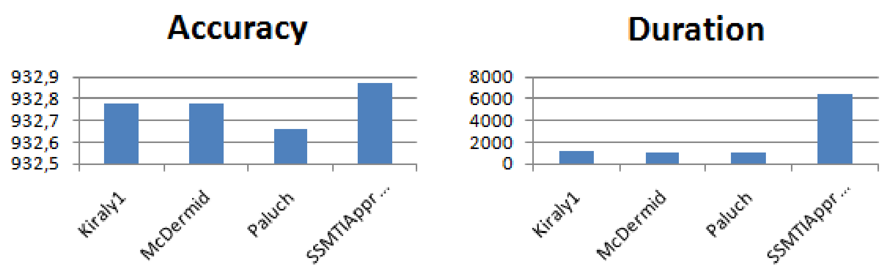

Figure 6.21: 100 instances ofl SSMTI of cardinality 1000.

Chapter 7

Conclusion

As a part of this thesis we implemented the following approximation algorithms for MAX SMTI problem:

- RandBrk
- ShiftBrk
- SSMTIApprox
- Kiraly
- McDermid
- Paluch
- GSA-LP

These algorithms are described and explained in Chapter 4 and some implemention remarks are summarized in Chapter 5. We compare all implemented algorithms with respect to its accuracy and duration. All results and observations are captured in Chapter 6.

According to our results we conclude that the best approximation algorithms for MAX SMTI are McDermid and Paluch with the approximation ratio of $\frac{3}{2}$. McDermid algorithm gives a little bit better results with respect to accuracy than Paluch, but Paluch runs faster on large inputs. On the other, side the worst results were given (except RandBrk) by IMY algorithm with the approximation ratio of 1.875. Although algorithms ShiftBrk and SSMTIApprox give good results, they are too slow to be used in practical applications.

As mentioned above, McDermid and Paluch compute a stable marriage with high accuracy, but both have the approximation ratio of $\frac{3}{2}$. An open problem is to find some worst-case example or to provide a better analysis of the approximation ratio.

Bibliography

[1] Canadian resident matching service. Available at: `http://www.carms.ca/eng/operations_algorithm_e.shtml`.

[2] lp_solve reference guide. Available at: `http://lpsolve.sourceforge.net/5.5/`.

[3] The national resident matching program. Available at: `http://www.nrmp.org/`.

[4] Scottish foundation allocation scheme. Available at: `http://www.dcs.gla.ac.uk/~pbiro/applications/uk_sfas.html`.

[5] Uriel G. Rothblum Alvin E. Roth and John H. Vande Vate. Stable matchings, optimal assignments, and linear programming. *Mathematics of Operations Research*, 18:803–828, 1993.

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

[7] Magnús M. Halldórsson, Kazuo Iwama, Shuichi Miyazaki, and Yasufumi Morita. Inapproximability results on stable marriage problems. In *LATIN '02: Proceedings of the 5th Latin American Symposium on Theoretical Informatics*, pages 554–568, London, UK, 2002. Springer-Verlag.

[8] Magnús M. Halldórsson, Kazuo Iwama, Shuichi Miyazaki, and Hiroki Yanagisawa. Randomized approximation of the stable marriage problem. *Theor. Comput. Sci.*, 325(3):439–465, 2004.

[9] Magnús M. Halldórsson, Kazuo Iwama, Shuichi Miyazaki, and Hiroki Yanagisawa. Improved approximation results for the stable marriage problem. *ACM Trans. Algorithms*, 3(3):30, 2007.

[10] Robert W. Irving. Stable marriage and indifference. In *CO89: Selected papers of the conference on Combinatorial Optimization*, pages 261–272, Amsterdam, The Netherlands, The Netherlands, 1994. North-Holland Publishing Co.

[11] Robert W. Irving and David F. Manlove. An 8/5 approximation algorithm for a hard variant of stable marriage. In *Proceedings of COCOON 2007: the 13th Annual International Computing and Combinatorics Conference*, volume 4598 of *Lecture Notes in Computer Science*, pages 548–558. Springer, 2007.

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

[13] Kazuo Iwama, Shuichi Miyazaki, David Manlove, and Yasufumi Morita. Stable marriage with incomplete lists and ties. In *ICAL '99: Proceedings of the 26th International Colloquium on Automata, Languages and Programming*, pages 443–452, London, UK, 1999. Springer-Verlag.

[14] Kazuo Iwama, Shuichi Miyazaki, and Naoya Yamauchi. A 1.875: approximation algorithm for the stable marriage problem. In *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 288–297, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[15] Shuichi Miyazaki Kazuo Iwama and Hiroki Yanagisawa. A 25/17-approximation algorithm for the stable marriage problem with one-sided ties. *Lecture Notes in Computer Science*, 6347, 2010.

[16] Zoltán Király. Better and simpler approximation algorithms for the stable marriage problem. In *ESA '08: Proceedings of the 16th annual European symposium on Algorithms*, pages 623–634, Berlin, Heidelberg, 2008. Springer-Verlag.

[17] David F. Manlove, Robert W. Irving, Kazuo Iwama, Shuichi Miyazaki, and Yasufumi Morita. Hard variants of stable marriage. *Theor. Comput. Sci.*, 276(1-2):261–279, 2002.

[18] D.F. Manlove. Stable marriage with ties and unacceptable partners. Technical Report TR-1999-29, Computing Science Department of Glasgow University, January 1999.

[19] Eric Mcdermid. A 3/2-approximation algorithm for general stable marriage. In *ICALP '09: Proceedings of the 36th International Colloquium on Automata, Languages and Programming*, pages 689–700, Berlin, Heidelberg, 2009. Springer-Verlag.

[20] Katarzyna Paluch. Faster and simpler approximation of stable matchings. *CoRR*, abs/0911.5660, 2009.

[21] H. Yanagisawa. *Approximation Algorithms for Stable Marriage Problems*. PhD thesis, Kyoto University, Graduate School of Informatics, 2007.

Appendix A

CD Content

The attached CD contains:

- applet.jar
  A Java applet that provides a web interface for all implemented approximation algorithms
- source.zip
  A source code of implemented algorithms
- instances.zip
  This zip file contains all instances that was used in algorithm comparisons. There are 20 folders which correspond to the appropriate figure, e.g.:
  folder fig_6_5 contains all instances considered in figure 6.5